Helmut Igelbach, Michael B. Zirpel

PC-FORTH©

Ein FORTH-Interpreter/Compiler für den PC-1500 mit Zusatzfunktionen zur Steuerung des MC-12-Systems

Inhaltsverzeichnis

Anhang

Vorwort

Die Sprache Forth ist in vielerlei Hinsicht revolutionär. Als relativ junge Sprache — sie wurde Anfang der 70'er Jahre in den USA entwickelt — erobert sie in Deutschland erst seit jüngster Zeit ein breiteres Publikum. Neuerdings finden Sie in fast allen Zeitschriften Beiträge über Forth.

FORTH ist mittlerweile für die meisten Rechner erhältlich. Die Forth-Implementationen reichen vom Homecomputer über fast alle Personalcomputer bis zum Minicomputer. Mit PC-FORTH ist die erste vollwertige Forth-Implementation für den PC-1500 gelungen, die sowohl die Geschwindigkeit als auch den Leistungsumfang eines Standard-Forth bietet. (Lediglich die Diskettenverwaltung über SCREEN- und BLOCK-Befehle ist nicht implementiert.)

FORTH-Programme sind portabel. Dank der Forth Interest Group (FIG) ist der FORTH-Kern gut standardisiert. PC-FORTH entspricht dem FIG-Standard, sodaß Sie die meisten Programme aus Zeitschriften oder Büchern direkt übernehmen können.

FORTH ist schnell. Je nach Anwendung ist PC-FORTH 5- bis mehr als 30mal so schnell wie das PC-1500 BASIC. Mit PC-FORTH ist der PC-1500 in vielen Anwendungen auch noch schneller als ein Apple II oder IBM PC in BASIC und erreicht damit eine neue Leistungsklasse.

FORTH erfordert sehr wenig Speicherplatz. Die meisten Compiler blähen die Programmlänge enorm auf, deshalb sind sie für Rechner mit weniger als 32K RAM kaum geeignet. Aufgrund der nahezu unglaublichen Codeeffizienz von Forth erreicht der PC-1500 auch hier eine neue Klasse.

FORTH ist strukturiert. Wie in anderen modernen Hochsprachen (PASCAL, C) enthält Forth schachtelbare Strukturen für Verzweigung und Schleifenbildung.

FORTH ist eine Interpretersprache. Daher können Programme genauso einfach interaktiv entwickelt werden, wie mit einem BASIC-Interpreter.

FORTH ist eine Compilersprache. Programme werden von Forth in einen sehr schnellen Zwischencode übersetzt, so daß die Ablaufgeschwindigkeit nah an manche konventionelle Compiler heranrückt.

FORTH ist maschinennah. Daher kann Forth in vielen Fällen als Ersatz zur Assemblerprogrammierung verwendet werden.

FORTH kann in beliebiger Richtung erweitert werden und so als problemorientierte Sprache für das jeweilige Anwendungsfeld optimiert werden.

FORTH ist eine offene Sprache. Da Forth dem Programmierer keine Zwangsjacken anlegt, können nahezu beliebige Daten- und Programmstrukturen verwirklicht werden.

Allerdings erfordert die Forth-Programmierung ein gründliches Umdenken. Am Anfang erscheint Forth vielleicht manchem als schwierig, gerade wenn er sich schon an andere Programmiersprachen gewöhnt hat. Ist man jedoch erst einmal mit den neuen Strukturen vertraut, zeigt es sich schnell, wie einfach und mächtig diese Sprache ist.

Diese Anleitung enthält neben den Befehlsbeschreibungen nur eine Kurzeinführung. Wem das nicht ausreicht, sei unser bald erscheinendes FORTH-Einführungsbuch empfohlen.

Forth eröffnet eine neue Dimension des Programmierens. Auch Sie werden von den Möglichkeiten dieser Sprache begeistert sein, wenn Sie sich eine Weile mit Forth beschäftigt haben.

Michael B. Zirpel

Gut Wildschwaige, Oktober 1984

Kapitel 1

1. Inbetriebnahme

Das PC-FORTH System besteht aus zwei Teilen: dem Forth-Code und einem BASIC-Teil, der für die Ein- und Ausgabe zuständig ist.

Dadurch ist einerseits gewährleistet, daß Forth-Programme zwischen Ein- und Ausgabe mit optimaler Geschwindigkeit ablaufen, andererseits kann die Ein- und Ausgabe leicht im BASIC geändert werden.

MC-12 FORTH läuft auf jedem MC-12 (A) System, bestückt mit einem PC-1500 oder PC-1500A und einem CE-161 Speichererweiterungsmodul.

1.1 Installation

Bauen Sie zunächst Ihr MC-12 System zusammen und vergessen sie nicht ein CE-161 in den PC-1500 einzusetzen.

Zur Installation gehen Sie folgendermaßen vor:

— Schalten Sie den PC-1500 in den PRO-Mode und geben Sie den Befehl

'NEW 0'

— Geben Sie den Befehl 'RVSLOAD'

— Schalten Sie den PC-1500 in den RUN-Mode

— Geben Sie 'RUN' ein um FORTH zu starten

Der Rechner meldet sich mit der Anzeige

PC—FORTH V1.1

Das MC-12 FORTH ist jetzt installiert. Um dies auszuprobieren, geben Sie folgendes ein (die vorhergehende Meldung wird jetzt überschrieben):

.CPU

wobei Sie die Eingabe mit ENTER abschließen. Daraufhin erscheint auf der Anzeige:

FH5801 OK

Das ist die Typenbezeichnung des im PC-1500 als CPU (Central Processing Unit) eingebauten Mikroprozessors, gefolgt von dem obligatorischen 'OK', mit dem Forth die Ausführung eines Kommandos bestätigt.

Der BASIC-Anfangszeiger steht bei &3800. Wenn Sie ein FORTH-Programm erstellt haben, dann speichern Sie den BASIC-Teil wie gewohnt ab. Beim Neuladen Ihres Programms geben Sie zunächst den Befehl 'RVSLOAD' ein, um das FORTH zu initialisieren. Den Standardbasicteil überschreiben Sie durch Nachladen Ihres Programms.

1.2 Abbruch und Wiederaufruf

Weil alle Ein- und Ausgaben des Forth-Systems über ein BASIC-Programm laufen, können Sie jederzeit anstelle einer Eingabe die BREAK-Taste betätigen und die Arbeit mit PC-FORTH genauso unterbrechen, wie Sie es von BASIC-Programmen gewohnt sind.

Für den Wiederaufruf von Forth gibt es drei Möglichkeiten:

Warmstart (DEF F)

Mit DEF F wird Forth erneut gestartet, wenn man es zuvor mit BREAK verlassen hat. Mit DEF F werden Sie normalerweise PC-FORTH aufrufen. (Dabei wird allerdings der Stack von Forth gelöscht, s.u.)

Kaltstart (DEF C oder RUN)

Mit DEF C oder RUN wird Forth gestartet, nachdem es von der Kassette geladen wurde.

Ansonsten dient der Kaltstart dazu, Forth zu initialisieren, und kann auch verwendet werden, um "aufzuräumen": Nach dem Kaltstart ist das Forth-System wieder in dem Zustand, in dem es von der Kassette geladen wurde. Insbesondere werden alle vom Anwender hinzugefügten Worte gelöscht.

Heißstart (DEF H oder GOTO"H")

Der Heißstart ist eine Spezialfunktion aus PC-FORTH und dient dazu, nach einem Sprung ins BASIC das unterbrochene FORTH-Programm an der Unterbrechungsstelle wieder fortzusetzen.

Wird manuell der Befehl DEF H erteilt, während Forth auf eine Eingabe wartet, so wird die letzte Eingabe automatisch wiederholt.

1.3 Ein- und Ausgabe

Die Ein- und Ausgabe läuft bei PC-FORTH über ein BASIC Programm. Dadurch können Sie, wie Sie es vom BASIC gewohnt sind, mit dem PC-FORTH System arbeiten.

Eingabe

Die Eingabe von Kommandos vollzieht sich bei PC-FORTH ähnlich, wie Sie es von BASIC-Programmen gewohnt sind, da alle Eingaben über einen INPUT-Befehl im BASIC-Teil des Forth-Systems laufen.

Allerdings ist die Eingabe so gestaltet, daß die zuletzt ausgegebene Meldung des FORTH-Systems in der Anzeige stehenbleibt und durch die neue Eingabe überschrieben wird. Durch Betätigung der ENTER-Taste ohne Eingabe kann die Anzeige aber jederzeit gelöscht werden.

Ausgabe über Anzeige

Die Anzeige des PC-1500 umfaßt nur 26 Zeichen. Deshalb wurden für die Anzeige von längeren Zeilen besondere Maßnahmen getroffen.

Werden mehrere oder längere Zeilen über die Anzeige ausgegeben, so geht Forth jedesmal, wenn die Anzeige gefüllt ist, in den Ausgabestop über. Durch Betätigung der ENTER-Taste kann die Ausgabe fortgesetzt werden. Das Zeichen '~' am Zeilenanfang kennzeichnet dabei, daß PC-FORTH noch nicht eingabebereit ist, sondern eine Ausgabe fortsetzt.

Beispiel:

Starten Sie PC-FORTH mit DEF F

Geben Sie das Kommando VLIST ein

Auf der Anzeige werden jetzt sämtliche Befehle von PC-FORTH ausgegeben. Jedesmal wenn Sie die ENTER-Taste betätigen, wird die nächste Zeile ausgegeben. Mit der Taste • können Sie VLIST abbrechen.

Mit den Tasten I und t können Sie während des Ausgabestops den TRACE ein- und ausschalten (vgl. die Befehle TRC, TRACE und TROFF).

Durch Anzeige des Cursors'__' auf der ersten Anzeigenposition zeigt PC-FORTH, daß es eingabebereit ist.

Ausgabe über CE-150

Sämtliche Ein- und Ausgaben von PC-FORTH können auf dem Drucker CE-150 protokolliert werden. Mit DEF L können Sie den Drucker ein- und ausschalten.

Geben Sie einmal mit eingeschaltetem Drucker das Kommando VLIST, Sie erhalten dann ein Listing des Vokabulars.

Ist der Drucker eingeschaltet, so führt PC-FORTH keine Ausgabestops mehr durch, sondern arbeitet kontinuierlich weiter.

Wenn Sie aber bei längeren Drucksequenzen wie VLIST eine beliebige Taste betätigen (Sie müssen die Taste u.U. eine Weile gedrückt halten), geht PC-FORTH in den Ausgabestop über. Sie können dann den TRACE einschalten, den Ausdruck wieder fortsetzen oder bei VLIST den Ausdruck mit • abbrechen.

Ausgabe über CE-158

Anstelle des Druckers CE-150 können Sie auch andere Drucker über die Schnittstelle CE-158 zur Ausgabe verwenden. Dazu unterbrechen Sie PC-FORTH mit BREAK und geben die entsprechenden BASIC-Kommandos, die zur LPRINT-Ausgabe an Ihr Gerät nötig sind (SETDEV PO).

Dann wird der externe Drucker genauso angesteuert wie das CE-150 und kann mit DEF L ein- und ausgeschaltet werden.

Kapitel 2

Einführung

In diesem Abschnitt werden die grundlegenden Eigenschaften von Forth in einem Schnellkurs behandelt. Dabei können natürlich nur die wichtigsten Punkte beleuchtet werden. Eine ausführliche Beschreibung aller Befehle von PC-FORTH finden Sie in Kapitel 3.

Auf den ersten Blick erscheint Forth vielleicht manchem Hochsprachen-Programmierer als zu elementar und maschinennah.

Forth ist aber auf Erweiterung ausgelegt: Man kann die Sprache in jeder Richtung beliebig komplex entwickeln, und das in ihren Compilereigenschaften genauso wie in den Lauf Zeitfunktionen. Fast beliebige Datentypen, Sprachstrukturen und Anwenderfunktionen können in Forth realisiert werden.

Sie werden nach einer gewissen Gewöhnungsphase schnell feststellen, daß Forth eine wesentlich mächtigere Programmiersprache als BASIC ist und auch andere Sprachen wie C in mancherlei Hinsicht übertrifft.

Forth erfordert jedoch ein gewisses Umdenken gegenüber herkömmlichen Programmiersprachen, da andere Grundkonzepte verwendet werden: Der Stack und die Postfixnotation, die Mischung aus einem Interpreter und einem Compiler, der selbst in Forth programmiert ist und ebenso wie das Runtime-Vokabular erweitert werden kann, das Wörterbuchkonzept und die unterschiedlichen Vokabularien sind für die meisten Hochsprachenprogrammierer genauso wie die Maschinennähe neu und ungewohnt.

2.1 Der Stack

Die Programmiersprache Forth verwendet, wie man es von manchen Taschenrechnern kennt, die sogenannte umgekehrte polnische Notation (UPN), manchmal auch als Postfixnotation bezeichnet.

Das bedeutet: Soll mit irgendwelchen Daten eine Operation durchgeführt werden, so gibt man erst die Daten, dann die Operation an.

Beispiel:

Es soll 123 + 5-17 berechnet werden.
In der UPN schreibt man:
123 5 + (123 + 5 = 128)
17— (128-17=111)

Damit das ganze vernünftig funktioniert, arbeitet Forth mit einem sogenannten Stack, zu deutsch 'Stapel' oder 'Keller' genannt.

Werden Zahlen eingegeben, so kommen sie auf den Zahlenstapel. Die zuletzt eingegebene Zahl liegt dann an oberster Stelle auf dem Stapel, die vorletzte eingegebene Zahl an zweitoberster Stelle usw.

Mit dem Befehl .S können Sie betrachten, was auf dem Stapel liegt, die unterste Zahl erscheint links, die oberste Zahl rechts.

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
DEFF	PC-FORTH 1.1	Warmstart von PC-FORTH
11 22 33	OK	Jetzt liegen die eingebenen Zahlen auf dem Stapel
.S	11 22 33 OK	33 liegt oben, 11 ganz unten

Wird eine Operation durchgeführt, dann holt sich die Operation die benötigten Daten vom Stapel und legt die Ergebnisse wieder auf dem Stapel ab.

Die Daten werden immer von oben vom Stapel genommen. Falls unter den benötigten Daten noch andere liegen, bleiben diese unverändert.

Bei der Addition mit + werden also die beiden obersten Zahlen vom Stapel genommen und addiert. Anschließend wird das Resultat der Addition auf den Stapel gelegt.

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
.S	11 22 33 OK	liegen noch auf dem Stapel
+ .S	11 55 OK	(33+22 = 22)
+ .S	66 OK	(55 + 11=66)

Zunächst werden Sie die Postfixnotation vielleicht als etwas schwierig empfinden, denn bei längeren Operationsfolgen erfordert sie ein gewisses Umdenken. Nach einer gewissen Eingewöhnungszeit dürfte sie Ihnen jedoch keine größeren Schwierigkeiten mehr bereiten.

Alle Operationen werden in Forth normalerweise beschrieben, indem man angibt, welche Daten sie auf dem Stapel erwarten und welche Daten sie auf dem Stapel hinterlassen. Beispielsweise wird die Operation + folgendermaßen beschrieben:

+	(nl n2--n3)	n3 = nl+n2

Dies bedeutet: + erwartet zwei Zahlen auf dem Stapel, nl und n2, die vom Stapel entfernt und durch n3, die Summe von nl und n2 ersetzt wird.

Man sieht also bei dieser Beschreibungsform immer auf einen Blick, was mit dem Stapel passiert.

Hier die wichtigsten arithmetischen Operationen aus Forth in der Kurzbeschreibung:

+	(nl n2 - n3)	n3 = nl+n2
-	(nl n2 - n3)	n3 = nl-n2
*	(nl n2 -- n3)	n3 = nl*n2
/	(nl n2 -- n3)	n3 = nl/n2
MOD	(nl n2 - n3)	n3 = Divisionsrest von nl/n2
ABS	(nl -- n2)	n2 = Absolutbetrag von nl
MINUS	(nl - n2)	n2 = -nl

Forth arbeitet mit ganzen Zahlen. Die meisten Operationen werden mit vorzeichenbehafteten 16-Bit Zahlen durchgeführt, der Wertebereich ist dann -32768... + 32767.

Wenn man längere Berechnungen in Forth durchführen will, so muß man ein wenig im voraus denken, aber man gewöhnt sich schnell daran.

Eine Reihe von Befehlen zur Stapelmanipulation erleichtert dabei die Arbeit.

DROP	(n ~)	löscht obersten Wert
DUP	(n - n n)	dupliziert den obersten Wert
SWAP	(nl n2 - n2 nl)	vertauscht die beiden obersten Werte
OVER	(nl n2 -- nl n2 nl)	kopiert zweiten Wert nach oben
PICK	(... nl -- ...n2)	kopiert den nl-ten Wert nach oben (Zählung ab nl mit 0)

Zur Ausgabe einer Zahl vom Stapel dient der Befehl. (also einfach ein Punkt). Dabei ist zu beachten, daß der ausgegebene Wert gleichzeitig vom Stapel gelöscht wird.

	(n ~)	Ausgabe des obersten Wertes

Die Verwendung des Stapels bietet mehrere Vorzüge: Man braucht zur Speicherung von Zwischenergebnissen keine Variablen, sondern kann die Werte auf dem Stapel hinterlassen. Die Übergabe von Werten an eigene Unterprogramme oder Funktionen findet einheitlich über den Stapel statt, genauso wie bei den Befehlen der Sprache Forth.

Forth verwendet noch einen weiteren Stapel für verschiedene (interne) Zwecke, den sog. Return-Stack (R-Stapel). Zur Unterscheidung wird der normale Stapel manchmal auch als C-Stapel (Computation-Stack) bezeichnet.

2.2 Variablen und Konstanten

Anders als in BASIC müssen Variablen vor der Verwendung definiert werden. Dazu dient das Wort VARIABLE, das in folgender Form benutzt wird:

n VARIABLE Name

Mit n wird der Anfangs wert der Variable vorgegeben, die unter dem angegebenen Namen im Forth-Wörterbuch abgelegt wird. Der Name kann dabei bis zu 31 Zeichen lang sein.

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
0 VARIABLE X	OK	X mit Anfangswert X = 0
5 VARIABLE Y1A	OK	Y1A mit Anfangswert Y1A = 5

Will man den Inhalt einer Variablen auf den Stapel holen, so geschieht das durch Aufruf des Variablennamens gefolgt von © (mit Leerzeichen dazwischen).

Soll ein Wert in einer Variablen gespeichert Verden, so erfolgt das durch Namensaufruf gefolgt von ! (Leerzeichen dazwischen).

Beispiel:

Eingabe:	**Anzeige:**	**Anmerkung:**
Y1A © .S	5 OK	Inhalt von Y1A auf den Stapel
7 + 3 * .S	36 OK	
X !	OK	in X speichern

Die Variablennamen werden bei der Definition in das Forth-Wörterbuch eingetragen, wobei gleichzeitig Speicherplatz für den Wert der Variablen reserviert wird. Wird ein Variablenname aufgerufen, so hinterläßt dieser Aufruf die Speicheradresse auf dem Stapel, bei der der Wert abgespeichert ist. Die Befehle © und ! arbeiten dann mit den Speicheradressen:

(a - n)	lädt die ab Adresse a gespeicherte Zahl n
(n a -)	speichert den Wert n ab Adresse a

Genauso wie Variablen können Konstanten definiert werden, deren Wert durch einfache Nennung des Namens auf den Stapel gebracht wird. Das geschieht mit der Anweisung:

n CONSTANT Name

die eine Konstante mit dem Wert n unter dem angegebenen Namen im Wörterbuch einträgt. Bei Aufruf des Namens wird der Wert n auf den Stapel gelegt.

2.3 Wortdefinitionen

Es gibt in Forth keine Unterscheidung zwischen Befehlen, Funktionen, Operatoren, Prozeduren oder Unterprogrammen. In Forth gibt es nur Worte, die weitgehend gleichberechtigt gebraucht werden.

Die vorhandenen Worte wie + oder DUP bilden den Grundwortschatz, aus dem der Anwender neue Worte bildet, die in das Forth-Wörterbuch völlig gleichberechtigt zu dem Grundvokabular aufgenommen werden.

Es gehört zur 'Philosophie' von Forth, daß Programme einfach aus der Definition neuer Forth-Worte bestehen. Jedes Programm hat somit automatisch einen Namen und ist ein neuer Befehl der Sprache Forth.

Die Wortnamen können aus beliebigen Zeichen gebildet werden, lediglich das Leerzeichen trennt einzelne Worte voneinander ab. Die maximale Namenslänge beträgt 31 Zeichen, die alle bei der Unterscheidung von Namen berücksichtigt werden.

Eine Wortdefinition hat folgende Form:

```
: Name Wortfolge ;
```

Damit wird das neue Wort 'Name' definiert und im Wörterbuch eingetragen. Die in der Definition angebene Wortfolge wird kompiliert und als Forth-Pseudocode im Wörterbuch direkt nach dem Namen abgelegt.

Wird später dann der Name aufgerufen, so wird die in der Definition angegebene Wortfolge ausgeführt.

Es ist gängige Forth-Praxis, zur Codeoptimierung auch elementare Befehlsfolgen durch neue Worte abzukürzen, wenn sie häufiger gebraucht werden.

Beispiel:

Definition des Wortes QU AD, das den obersten Stapel wert quadriert.
Eingabe (Leerzeichen nicht vergessen, am Ende der Eingabe ENTER):

```
: QUAD DUP * ;
```

Jetzt ist QU AD definiert und kann verwendet werden:
3 QU AD . liefert jetzt das Ergebnis 9
QU AD kann aber auch in neuen Definitionen verwendet werden, z.B. in der Definition des Wortes CUBE, das den obersten Stapelwert mit 3 potenziert:

```
: CUBE  DUP  QUAD  *  ;
```

```
3  CUBE  .  liefert das Ergebnis 27
```

Sie können in Forth jederzeit Worte des Grundvokabulars mit neuen Namen versehen, z.B. um Abkürzungen zu verwenden oder Forth einzudeutschen.

Beispiel:

```
:  PRINT  .  ;     ( PRINT anstelle von .)
:  D  DUP  ;       ( D anstelle von DUP )
```

Auch können Sie unter dem Namen von bereits definierten Worten neue Worte definieren. Dann erscheint zwar die Meldung "~ neu", aber die Definition wird akzeptiert.

Beispiel:

```
:  .  DUP  .  ;    ( nichtlöschende Zahlenausgabe )
```

Forth nimmt immer die letzte Definition eines Namens als augenblicklich gültige Definition an.

Es gehört zur 'Philosophie' von Forth, die Wortdefinitionen möglichst einfach und kurz zu halten.

Soll ein längeres Programm realisiert werden, so wird man dieses Programm nicht in eine einzige Wortdefinition packen, sondern eine möglichst weitgehende Zerlegung des Programms in sinnvolle Teile durchführen. Zu jedem dieser Teile wird ein Wort definiert. Jedes dieser Worte testet man zunächst ehe man weitere Worte definiert. Zum Schluß kann dann ein 'Hauptwort' definiert werden, daß die 'Unterworte' zusammenfaßt.

Forth bietet auf einfache und elegante Weise die Möglichkeit, in komplexeren Programmsystemen eigene Kommandosprachen zu realisieren: Man definiert für die einzelnen Operationen entsprechende Worte, die genauso wie andere Forthworte aufgerufen werden können. Dabei spart man sich die Programmierung eines Hauptprogramms, das Einzeloperationen über Menüs oder Kommandos aufruft. Das Forth-System erledigt das ganz von allein und bietet gleichzeitig wieder die Möglichkeit, in der neuen Kommandosprache zu programmieren.

2.4 Editieren der Programme

Im Forthinterpreterbetrieb können Sie jederzeit neue Worte definieren, wie das in den bisherigen Beispielen geschehen ist. Da die entsprechenden Programme gleich kompiliert werden, ist es nicht mehr möglich, sie zu editieren, um z.B. eine Einfügung vorzunehmen. Die einzige Editiermöglichkeit im Forthinterpreterbetrieb bietet der Befehl FORGET, der in der Form:

    FORGET Name

angewendet wird. Er bewirkt, daß alle Definitionen, die nach der neuesten Definition des angegebenen Namens erfolgt sind (inkl. Name) gelöscht werden.

Man kann dann die geänderten Definitionen nocheinmal eintippen.

Weil das für längere Programme natürlich nicht ausreicht, bietet Forth die Möglichkeit, Programmtexte von einem Editor zu übernehmen und zu kompilieren.

Das Standard FIG-Forth benutzt dazu sogenannte SCREEN's. Dabei handelt es sich um Textspeicher auf der Diskette, die stets 16 Zeilen von 64 Zeichen umfassen und mit dem SCREEN-Editor beschrieben werden können. Diese SCREEN's sind durchnumeriert und können gezielt vom Forthinterpreter angesprochen werden.

PC-FORTH geht einen anderen Weg, indem es den BASIC-Programmspeicher und den BASIC-Editor anstelle der SCREEN's verwendet. Dadurch haben die PC-FORTH Anwender den Vorteil eines komfortableren Editors und außerdem weitergehende Möglichkeiten BASIC und Forth zu mischen (s.u.).

Um Forth-Quellprogramme einzugeben, schaltet man (nach Verlassen des Forth-Interpreters mit BREAK) in den PRO-Mode des PC-1500 und schreibt die Forth-Befehle in normale BASIC-Programmzeilen, bei denen nach der Zeilennummer Anführungsstriche eingegeben werden.

Beispiel:

```
200": QU AD ( n -- n*n ) DUP * ;
210": CUBE ( n--n*n*n ) DUP QUAD * ;
```

Wie Sie im Beispiel sehen können, werden die Klammern in Forth verwendet, um Kommentare ins Programm aufzunehmen. Der öffnenden Klammer muß unbedingt ein Leerzeichen folgen, da es sich um ein Forth-Wort handelt. Forth ignoriert alles, was nach einer öffnenden Klammer steht, bis zur nächsten schließenden Klammer oder Zeilenende.

Bei der Eingabe von Forth-Quellprogrammen dürfen Sie allerdings nicht die zum Forth-System gehörenden BASIC-Zeilen ändern oder löschen. Diese Zeilen haben Nummern, die kleiner als 200 sind.

Nach dem eingegebenen Forth-Quellprogramm folgt eine Zeile mit dem Wort P> , das das Ende des Forth-Quelltextes markiert.

Beispiel:

```
220" P>
```

Um die Quelltexte zu kompilieren, wird PC-FORTH gestartet und dann die Zeilennummer, bei der der Forth-Quelltext beginnt, eingegeben und anschließend der Befehl <P.

Beispiel:

```
200  <P
```

Mit diesem Befehl wird einfach die Forth-Eingabe umgeleitet. Statt von der Tastatur erfolgt die Eingabe ins Forth aus dem BASIC-Programmspeicher. Der Befehl P> schaltet dann wieder auf die Tastatureingabe zurück.

Dieses Einlesen aus dem BASIC-Programmspeicher benötigt einige Zeit. Ist es abgeschlossen, so werden etwaige Meldungen und anschließend das obligatorische OK ausgegeben.

Forth-Quellprogramme können wie gewöhnliche BASIC-Programme editiert oder auf Kassette gespeichert werden. An den für das Forth-System benötigten BASIC-Zeilen sollten Sie sich dabei nicht stören und sie jeweils mit auf der Kassette speichern.

2.5 Kontrollstrukturen

Forth unterstützt die strukturierte Programmierung durch schachtelbare Strukturen für Schleifenbildung und Verzweigung, genauso wie es in anderen modernen Programmiersprachen üblich ist.

Diese Strukturen dürfen allerdings nur innerhalb von Wortdefinitionen (Programmen) verwendet werden.

Bedingungen werden bei den Strukturworten ebenfalls in der Postfixnotation angegeben: Erst kommt die Bedingung, dann der jeweilige Kontrollbefehl (z.B. IF).

Zur Formulierung der Bedingungen stehen eine Reihe von Vergleichsbefehlen zur Verfügung, die je nach Ausgang des Vergleichs auf dem Stapel eine 0 oder 1 hinterlassen.

Die wichtigsten Vergleichsbefehle sind:

<	(nl n2 -- n3)	n3 = 1 wenn nl<n2, sonst n3 = 0
=	(nl n2 -- n3)	n3 = 1 wenn nl = n2, sonst n3 = 0
>	(nl n2 -- n3)	n3 = 1 wenn nl >n2, sonst n3 =0
0<	(nl--n2)	n2=l wenn nl<0, sonst n2 = 0
0=	(nl-n2)	n2=l wenn nl=0, sonst n2 = 0

Der Befehl 0 = wird oft dazu verwendet eine Bedingung zu invertieren und wird deshalb in manchen Forth-Dialekten auch mit NOT bezeichnet. (Wenn Sie das ebenfalls wollen, können Sie mit : NOT 0= ; den NOT-Befehl in PC-FORTH einreihen. Die restlichen Vergleiche wie > = können Sie ebenfalls in einfacher Weise realisieren : > = < NOT ; usw.)

Beispiel:

X © 3 > (Test, ob der Inhalt der Variablen X > 3)

Bedingungen können mit den zur Verfügung stehenden logischen Befehlen verknüpft werden.

AND	(nl n2 -- n3)	n3 = nl AND n2 (Bitweise Und)
OR	(nl n2 -- n3)	n3 = nl OR n2 (Bitweise Oder)
XOR	(nl n2 -- n3)	n3 = nl XOR n2 (Bitweise Exklusiv-Oder)

Beispiel:

X © 3 > X © 1 0 < AND (Test, ob X>3 und X<10)

Die Kontrollstrukturen veranlassen beim Programmablauf Verzweigungen im Programm, je nachdem ob Bedingungen erfüllt sind oder nicht.

Die Bedingungen werden dabei immer vom Stapel genommen: der Wert 0 kennzeichnet, daß die Bedingung nicht erfüllt ist, jeder andere Wert, daß die Bedingung erfüllt ist.

Folgende Strukturbefehle sind in Forth vorhanden:

Bedingung IF Wortfolge ENDIF

Die Wortfolge zwischen IF und ENDIF wird nur dann durchlaufen, wenn der bei IF vom Stapel genommene Wert ungleich Null ist.

Bedingung IF Wortfolgel ELSE Wortfolge2 ENDIF

Die Wortfolgel wird durchlaufen, wenn der bei IF vom Stapel genommene Wert ungleich Null ist, andernfalls die Wortfolge2. In beiden Fällen wird das Programm nach ENDIF fortgesetzt.

BEGIN Wortfolge Bedingung UNTIL

Die Wortfolge zwischen BEGIN und UNTIL wird durchlaufen. Ist der bei UNTIL vom Stapel genommene Wert gleich Null, so wird die Wortfolge erneut durchlaufen, andernfalls wird das Programm hinter UNTIL fortgesetzt.

BEGIN Wortfolgel Bedingung WHILE Wortfolge2 REPEAT

Wortfolgel wird durchlaufen. Ist der bei WHILE vom Stapel genommene Wert ungleich Null, so wird mit Wortfolge2 fortgesetzt und anschließend Wortfolgel wiederholt. Ist bei WHILE der vom Stapel genommene Wert gleich Null, so wird der Programmlauf hinter REPEAT fortgesetzt.

Eine Zählschleifenstruktur (ähnlich FOR...NEXT) steht in Forth ebenfalls zur Verfügung. Es handelt sich dabei um die Worte DO und LOOP bzw. + LOOP, die folgendermaßen verwendet werden:

nl n2 DO Wortfolge LOOP

DO nimmt die Werte nl und n2 vom Stapel. Der Schleifenzähler wird auf den Anfangswert n2 gesetzt, anschließend wird die Wortfolge durchlaufen. Bei LOOP wird der Schleifenzähler um 1 erhöht und mit der Schranke nl verglichen. Ist die Schranke erreicht oder überschritten, so wird das Programm hinter LOOP fortgesetzt, andernfalls wird die Wortfolge zwischen DO und LOOP wiederholt.

Soll nicht in Einer-Schritten gezählt werden, verwendet man anstelle von LOOP den Befehl + LOOP folgendermaßen:

nl n2 DO Wortfolge n3 +LOOP

+ LOOP nimmt den Wert n3 vom Stapel und addiert ihn zum Schleifenzähler hinzu, ansonsten funktioniert das ganze wie bei DO und LOOP.

Beispiel:

: TEST 10001 1 DO LOOP ;
Diese leere Schleife wird lOOOOmal durchlaufen und kann als Benchmarktest dienen. Wenn Sie das Wort TEST aufrufen, dauert die Ausführung ca. 4 Sekunden. (Gegenüber ca. 140 Sekunden bei FOR-NEXT im BASIC ist PC-FORTH darin also 35mal schneller.)

Alle Schleifen- und Verzweigungsstrukturen können ineinander verschachtelt werden, wie es den Regeln der strukturierten Programmierung entspricht.

2.6 Ein-/Ausgabebefehle

Die Ein- und Ausgabe von Zahlen kann in Forth in jedem beliebigen Zahlensystem erfolgen. Mit den Befehlen HEX und DEC können Sie ins Hexadezimalsystem umschalten bzw. ins Dezimalsystem zurückschalten.

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
HEX	OK	Hexadezimalsystem
14 8 + .	IC OK	Ergebnis hexadezimal
FF DEC .	255 OK	Wieder dezimal

Andere Zahlensysteme können gewählt werden, indem man in die Systemvariable BASE den Wert der Zahlensystembasis speichert. 16 BASE ! ist z. B. identisch zu HEX, mit 2 BASE ! können Sie auf das Dualsystem umschalten.

Forth versucht jede Eingabe zunächst als Wort zu interpretieren. Deshalb ist es u.U. nötig, daß man z.B. Hex-Zahlen durch führende Nullen kennzeichnet, um Verwechselungen mit Worten zu vermeiden. Die Hex-Zahl DEC muß z.B. als ODEC eingegeben werden, denn DEC wird als Befehl aufgefaßt.

Folgende Befehle stehen zur Zahlenausgabe zur Verfügung:

	(n —)	Standardausgabe einer Zahl
.R	(nl n2 —)	gibt nl rechtsbündig in einem Feld der Länge n2 aus

Neben den bisher verwendeten vorzeichenbehafteten 16-Bit Zahlen kennt Forth noch vorzeichenfreie 16-Bit Zahlen. Der Wertebereich beträgt dann 0...65535, als Abkürzung wird für solche Zahlen a (Adresse) oder u (unsigned) verwendet.

Mit folgendem Befehl können Sie vorzeichenfreie Zahlen ausgeben:

U.	(a -)	Ausgabe einer vorzeichenfreien Zahl

Außerdem enthält Forth noch die doppeltlangen (doppeltgenauen) 32-Bit Zahlen, bei denen der Wertebereich -2147483648... +2147483647 beträgt. Diese Zahlen werden mit d abgekürzt.

Solche doppeltlangen Zahlen gibt man ein, indem man bei der Zahleneingabe einen Dezimalpunkt verwendet. Ausgegeben werden Sie mit den Befehlen:

D.	(d -)	Ausgabe einer doppeltlangen Zahl d
D.R	(d n -)	Ausgabe von d rechtsbündig im Feld der Länge n

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
1234567.	OK	Eingabe einer Doppelzahl
D.	1234567	Ausgabe einer Doppelzahl

Diese doppeltlangen Zahlen belegen gegenüber einfachen Zahlen zwei Stapelpositionen (4 Bytes gegenüber 2 Bytes). Nach Forth-Konvention liegt die höherwertige Hälfte oben auf dem Stapel.

Sie finden im Vokabular in Kapitel 3 eine ganze Reihe von Befehlen zur Behandlung der Doppelzahlen, z.B. 2DUP, D + , DABS. Allerdings fehlen manche Worte im Vergleich zu einfachen Zahlen und müssen vom Anwender definiert werden, falls er sie benötigt.

Die Position des mit Doppelzahlen eingegebenen Dezimalpunktes, der an beliebiger Stelle innerhalb der eingegebenen Zahl stehen darf, kann vom Programmierer der Systemvariablen DPL entnommen werden, so daß die Programmierung von Festkommazahlen möglich ist.

Texte werden mit dem Befehl ," ausgegeben, das Textende wird mit" markiert. Die Textausgabe erfolgt also in der Form:

." auszugebender Text"

(In PC-FORTH wird aus technischen Gründen anstelle von " immer" verwendet.)

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
: TEST1 ." PROBETEXT " ;	OK	
TEST1	PROBETEXT OK	Ausgabe gefolgt von OK

Alle Ausgaben aus Forth kommen zunächst in einen Puffer. Erst wenn dieser voll ist oder wenn der Befehl CR erfolgt, wird der Puffer tatsächlich ausgegeben.

CR (—) Ausgabe des Puffers und Beginn einer neuen Zeile

Beispiel:

Eingabe:	Anzeige:	Anmerkung:
: TEST2 11 . CR		Testdefinition
22 . 33 . CR ;	OK	
TEST2	11	Fortlaufende Ausgabe, jeweils
	22 33	neue Zeile, wenn Drucker
	OK	an, sonst Ausgabestop
		nach jeder Zeile

Folgende weitere Befehle stehen für die Ausgabe zur Verfügung:

EMIT	(n --)	gibt das Zeichen mit dem ASCII-Code n aus
TYPE	(a n -)	gibt n Zeichen aus, die ab Adresse a gespeichert sind
SPACE	(-)	gibt ein Leerzeichen aus
SPACES	(n --)	gibt n Leerzeichen aus

Zur programmgesteuerten Eingabe stehen folgende Befehle in PC-FORTH zur Verfügung:

EXPECT	(a n -)	Eingabe von maximal n Zeichen, die ab Adresse a gefolgt von einer 0 gespeichert werden.
KEY	(- n)	Eingabe eines Zeichens, liefert seinen ASCII-Code n

Zur Umwandlung von eingegebenen Zeichen in eine Zahl steht die Funktion NUMBER zur Verfügung (entspricht dem VAL aus BASIC):

NUMBER (a ~ d) wandelt die ab Adresse a gespeicherte Zeichenfolge in eine Doppelzahl um.

Die mit EXPECT eingegebenen Zeichenfolgen müssen natürlich irgendwo gespeichert werden. Forth stellt dafür einen Speicherbereich oberhalb des Wörterbuchs zur Verfügung, dessen Lage sich allerdings nach jedem Eintrag ins Wörterbuch verschiebt. Der Befehl PÄD liefert die augenblickliche Startadresse dieses Speicherbereichs.

Zur Speicherung von einzelnen Zeichen kann man mit Forth auch einzelne Speicherzellen oder Speicherbereiche ansprechen:

C©	(a - n)	lädt den Inhalt n der Speicherzelle a auf den Stapel
C!	(n a -)	speichert n (n = 0...255) bei Adresse a.
ERASE	(a n -)	löscht n Bytes ab Adresse a (mit Nullen)
BLANKS	(a n -)	füllt n Bytes ab Adresse a mit Leerzeichen
FILL	(a nl n2 -)	füllt nl Bytes ab a mit dem Wert n2 (n2 = 0...255)
CMOVE	(al a2 n -)	kopiert n Bytes ab Adresse al nach a2

Werden Texte gespeichert, so wird der Zeichenfolge ein Längenbyte vorangestellt, das die Anzahl der gespeicherten Zeichen enthält oder der Zeichenfolge werden eine oder mehrere Nullen als Endzeichen angehängt.

2.7 PC-FORTH und BASIC

PC-FORTH ist auf die Zusammenarbeit mit dem PC-1500 BASIC ausgerichtet, wie bereits die EinVAusgabe und die Verwendung des BASIC-Editors zeigen.

Sie können aus einem Forth-Programm heraus ein BASIC-Programm starten. Dies leistet der Befehl BAL:

BAL (n -) springt in die BASIC-Zeile mit der Nummer n (n=1...255)

(

Mit diesem Befehl können also nur Zeilen mit einer Nummer aus dem Bereich 1...255 aufgerufen werden. Über einen Zwischensprung kann der Radius aber leicht erweitert werden.

Der Rücksprung ins FORTH erfolgt mit der BASIC-Anweisung GOTO"H", das unterbrochene Forth-Programm wird dann nach der Unterbrechungsstelle fortgesetzt.

Im BASIC dürfen allerdings die Variablen O, S, L, I$(0) und O$(0) nicht verändert werden, da sie von FORTH benötigt werden.

/

Sie können in einem Forth-Programm BASIC-Textzeilen ausgeben. (BASIC-Textzeilen beginnen mit Anführungstrichen und enthalten keine BASIC-Befehle).

.LINE	(n -)	gibt die BASIC-Textzeile mit der Nummer n aus.

PC-FORTH verwendet diesen Befehl bei der Ausgabe der Fehlermeldungen, die in BASIC-Zeilen gespeichert sind.

Für Manipulationen an BASIC-Zeilen (z.B. RENUMBER) liefert der Befehl B# die Speicheradresse, bei der der Text einer BASIC-Zeile beginnt.

Daten können aus BASIC-Variablen mit dem Befehl ©: übernommen werden und mit dem Befehl !: in BASIC-Variablen gespeichert werden.

Diese Befehle werden in der Form:

©: "Variablenname" bzw. !: "Variablenname"

verwendet. Der Variablenname ist dabei in Anführungstriche zu setzen. Diese Befehle können allerdings nur über BASIC-Zeilen und nicht direkt von der Tastatur eingelesen werden. Die genaue Beschreibung dazu finden Sie in Kapitel 3.

2.8 Spracherweiterungen

Eigentlich ist jedes neudefinierte Wort eine Erweiterung der Sprache Forth: Man kann es direkt eingeben und ausführen lassen, man kann es aber auch innerhalb einer weiteren Definition verwenden, wobei der Aufruf des neuen Wortes kompiliert wird.

Will man jedoch den Forth-Compiler, der die in den Definitionen angegebenen Wortfolgen übersetzt, durch neue Worte erweitern oder neue Variablentypen definieren, so sind verschiedene Dinge zu berücksichtigen, die jetzt kurz beleuchtet werden. (In Anhang C finden Sie ausführlichere Erklärungen über die Arbeitsweise des Compilers und den Aufbau des Wörterbuchs.)

Bei einer Definition wird zunächst der Name des neudefinierten Wortes und einige Kennzeichnungen in das Wörterbuch eingetragen. Anschließend folgt das sogenannte Parameterfeld, in das die in der Definition angegebene Wortfolge kompiliert gespeichert wird.

Eine Reihe von Befehlen steht zur Verwaltung des Wörterbuchs zur Verfügung. Um einzelne Bytes in das Wörterbuch zu schreiben, verwendet Forth u.a. folgende Worte:

HERE	(— a)	liefert die Adresse des ersten freien Bytes hinter den bisherigen Einträgen
,	(n --)	trägt n (2 Bytes) ab HERE ins Wörterbuch ein und erhöht HERE um 2
C,	(n -)	trägt n (1 Byte) bei HERE ein und erhöht HERE um 1
ALLOT	(n -)	läßt n Bytes im Wörterbuch frei (erhöht HERE um n)

Diese Befehle benötigen Sie z.B. wenn Sie Variablen eines neuen Datentyps (z.B. Felder oder Textvariablen) kreieren wollen und dafür Speicherplatz im Wörterbuch verwenden.

Zur Definition neuer Datentypen werden die Worte <BUILDS und DOES> verwendet und zwar in folgender Form:

```
: Typname <BUILDS Wortfolge 1 DOES> Wortfolge2 ;
```

Der angegebene Typname wird damit zu einem Definitionswort (wie VARIABLE oder CONSTANT) gemacht.

Will man ein Objekt des neuen Typs definieren, so geschieht das in der Form:

```
Typname    Objektname
```

Dabei wird die Wortfolge zwischen <BUILDS und DOES> aus der Typdefinition ausgeführt.

Wenn man ein so definiertes Objekt aufrufen will, so geschieht das wie gewohnt durch Angabe des Objektnamens:

Objektname

Dabei wird die Wortfolge ausgeführt, die in der Typdefinition nach DOES> steht, wobei die Adresse des Parameterfeldes (PFA) des Objekts auf dem Stapel übergeben wird.

Beispiel:

Zur Speicherung von doppeltlangen Zahlen stehen zwar die Befehle 2! und 2© zur Verfügung, aber keine Doppelvariablen, deren Definition nachgeholt werden kann:

```
: 2VARIABLE          ( definiert den Typ Doppelvariable)
  <BUILDS
  HERE 2!            ( nimmt Initialisierungswert und speichert ihn ab
                     HERE)
 4 ALLOT             ( der Wert der Variablen belegt 4 Bytes im
                     Wörterbuch)
  DOES> ;            ( hinterläßt Parameterfeldadresse, macht sonst
                     nichts)
```

Anwendung von 2VARIABLE:

```
0. 2VARIABLE A       ( definiert die Variable A mit dem Anfangswert 0)
12345678. A 2!       ( speichert in A den Wert 12345678)
A 2© D.              ( gibt den Inhalt von A aus)
```

Analog kann der Typ 2CONSTANT definiert werden:

```
: 2CONSTANT          ( definiert den Typ Doppelkonstante)
  <BUILDS
```

```
HERE 2!          (trägt den Wert ein)
4 ALLOT
DOES>
2© ;             (lädt den Wert auf den Stapel)
```

Mit <BUILDS und DOES> können Sie die verschiedensten Datentypen oder -strukturen auf elegant einfache Weise implementieren, die zugehörigen Operationen werden dann als Worte definiert.

Dabei ist es allerdings manchmal nötig, den Compiler zu beeinflussen, was in Forth jedoch keine größeren Schwierigkeiten verursacht.

Wenn Sie eine Definition eingeben, werden normalerweise alle eingegeben Worte kompiliert und erst später beim Aufruf ausgeführt.

Beispiel:

```
: TEST  HEX 7750  C©  .  ;
```

Wird dieser Befehl ausgeführt, so schaltet Forth ins Hexadezimalsystem um und gibt den Inhalt der Speicherzelle 7750 aus. War während der Eingabe der Definition das dezimale Zahlensystem eingestellt, so ist mit 7750 dezimal 7750 gemeint, also hexadezimal 1D4C, und der entsprechende Zahlenwert wurde kompiliert. TEST zeigt dann also den Inhalt der Speicherzelle hex 1D4C an.

Es gibt jedoch die Möglichkeit, auch während einer Definition Befehle auszuführen. Dazu wird mit ((die Kompilation unterbrochen und mit)) wieder aufgenommen.

Beispiel:

```
: TEST  ((  HEX  ))  7750  C©  .  ;
```

Jetzt wird während der Definition auf das Hexadezimalsystem umgeschaltet und 7750 als hex 7750, also dezimal 30544, kompiliert. Während des Aufrufs von TEST wird das eingestellte Zahlensystem überhaupt nicht verändert.

Mit Hilfe von ((und)) kann während einer Definition auch eine Berechnung durchgeführt werden, beispielsweise um das Ergebnis mit dem Befehl LITERAL als Konstante in das Programm aufzunehmen:

Beispiel:

```
: TEST DUP 5  +  *  ;            ( n -- n*(n + 5)
```

Die Zahl 5 ist als Konstante im kompilierten Programn enthalten;
sie kann aber auch während der Kompilierung errechnet werden, ohne das kompilierte Programm zu ändern:

```
: TEST DUP  ( ( 2 3 +  ))  LITERAL  +  *  ;
```

Sollen also während einer Definition die Befehle ausgeführt und nicht kompiliert werden, so verwendet man ((und)).

Es gibt aber auch Worte in Forth, die immer sofort ausgeführt werden. Man nennt sie Immediate-Worte. Solche Immediate-Worte werden, auch wenn sie innerhalb einer Definition verwendet werden, unmittelbar ausgeführt, ohne daß dazu ((und)) verwendet werden müssen. Zu diesen Immediate-Worten gehört z.B. das Semikolon ; oder ((.

Wenn Sie eigene Immediate-Worte definieren wollen, so geschieht das mit dem Befehl IMMEDIATE, der direkt nach Beendigung der Definition eines Wortes gegeben wird, um das Wort als immediate zu kennzeichnen.

Will man allerdings Immediate-Worte wie z.B. das Semikolon in einer Definition kompilieren, so muß man das Wort ((COMPILE)) voranschicken.

Beispiel:

```
: DEF ((COMPILE)) : ;          ( anstelle von : jetzt DEF verwendbar)
: ENDDEF ((COMPILE)) ; ;       ( anstelle von ; jetzt ENDDEF verwendbar)
IMMEDIATE                      ( ENDDEF muß natürlich wie ; immediate
                               sein)
```

Eine andere Kategorie bilden die Forth-Worte, die nur innerhalb von Definitionen verwendet und dabei unmittelbar (IMMEDIATE) ausgeführt werden, aber gleichzeitig die Kompilierung anderer Worte veranlassen.

Sämtliche Strukturbefehle wie IF oder ENDIF gehören zu dieser Kategorie: Es handelt sich bei ihnen um IMMEDI ATE-Worte. Während der Kompilierung prüfen sie anhand von Werten, die sie auf dem Stapel hinterlassen bzw. vom Stapel nehmen, die Struktur des Programms. In das Programm werden sie jedoch nicht aufgenommen, sondern stattdessen veranlassen sie, daß entsprechende Sprungbefehle in das Programm kompiliert werden. Dazu verwenden sie den Befehl COMPILE.

Beispiel:

```
: HEXA HEX COMPILE HEX ; IMMEDIATE

: TEST HEXA 7750 © . ;
```

Da HEXA ein Immediate-Befehl ist, wird auch innerhalb einer Definition mit HEXA auf das Hexadezimalsystem umgeschaltet. Gleichzeitig veranlaßt HEXA die Kompilierung von HEX, sodaß die Umschaltung in das Hexadezimalsystem auch in das kompilierte Programm aufgenommen wird und daher beim Aufruf von TEST ebenfalls durchgeführt wird.

2.9 Vokabularien

Bisher wurden die definierten Worte einfach in das Wörterbuch eingetragen, ohne daß festgelegt wurde, in welchen Teil des Wörterbuchs sie aufgenommen werden sollten.

Im Wörterbuch kann man aber verschiedene Vokabularien definieren und so ganze Wortgruppen unter einem Namen zusammenfassen. Am Anfang gibt es allerdings nur ein einziges Vokabular in PC-FORTH, das den Namen FORTH trägt und deckungsgleich mit dem ganzen Wörterbuch ist.

Die Definition eines Vokabulars geschieht in folgender Weise:

```
VOCABULARY Vokabularname IMMEDIATE
```

Damit wird ein neues Vokabular unter dem angegebenen Name angelegt. Sollen Definitionen in dieses neue Vokabular eingetragen werden, so geschieht das mit dem Aufruf des Vokabularnamens gefolgt von DEFINITIONS:

Vokabularname DEFINITIONS

Alle folgenden Wortdefinitionen werden jetzt in das aufgerufene Vokabular eingetragen.

Sollen Definitionen wieder in das Grundvokabular FORTH eingetragen werden, so gibt man ein:

FORTH DEFINITIONS

Das Vokabular, in dem die jeweiligen Definitionen eingetragen werden, heißt CURRENT-Vokabular.

Soll ein Vokabular aktiviert werden, damit die darin definierten Worte für die Anwendung zur Verfügung stehen, so geschieht dies einfach mit dem Aufruf des Vokabularnamens:

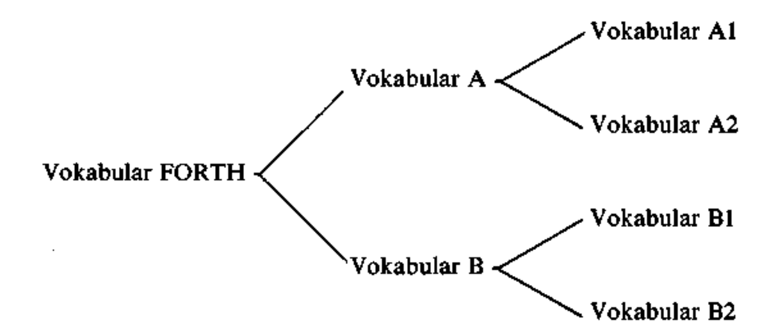
Vokabularname

Alle eingegebenen Worte werden dann zuerst im aufgerufenen Vokabular gesucht. Werden sie dort gefunden, so werden sie ausgeführt bzw. kompiliert. Werden sie dort nicht gefunden, so geht die Suche in dem Vokabular weiter, in dem das aufgerufene Vokabular definiert wurde. Dadurch stehen die Worte des FORTH-Vokabulars immer zur Verfügung.

Das gerade aktive Vokabular, in dem die eingegebenen Worte gesucht werden, heißt CONTEXT-Vokabular.

Je nach Reihenfolge der Definition von Vokabularien, können baumartige Strukturen für die Wortsuche definiert werden.

Beispiel:



Wird erzeugt durch:

```
FORTH DEFINITIONS                         (Anfangszustand)
Eintrag von Definitionen in FORTH
VOCABULARY A IMMEDIATE                    (Definition des Vokabulars A)
  A DEFINITIONS                           (A als CURRENT-Vokabular)
  Eintrag von Definitionen in A
  VOCABULARY AI IMMEDIATE                 (Definition des Vokabulars AI)
    AI DEFINITIONS                        (AI als CURRENT-Vokabular)
    Eintrag von Definitionen in AI
  A DEFINITIONS                           (A als CURRENT-Vokabular)
  VOCABULARY A2 IMMEDIATE                 (Definition des Vokabulars A2)
    A2 DEFINITIONS                        (A2 als CURRENT-Vokabular)
    Eintrag von Definitionen in A2
FORTH DEFINITIONS                         (Anfangszustand)
Weitere Einträge in FORTH
VOCABULARY B IMMEDIATE                    (Definition des Vokabulars B)
  B DEFINITIONS                           (B als CURRENT-Vokabular)
  Eintrag von Definitionen in B
  VOCABULARY Bl IMMEDIATE                 (Definition des Vokabulars Bl)
    Bl DEFINITIONS                        (Bl als CURRENT-Vokabular)
    Eintrag von Definitionen in Bl



                                          (Anfangszustand)
```

Nach:

AA2

können die Worte des Vokabulars A2 aufgerufen werden.

Nach:

A A2 DEFINITIONS

können weitere Definitionen in das Vokabular A2 aufgenommen werden.

Typische Anwendungsbeispiele in Forth-Systemen für solche Vokabularien sind das EDITOR-Vokabular, mit denen die Befehle des SCREEN-Editiors aktiviert werden, oder das ASSEMBLER-Vokabular, mit dem Maschinenbefehle in der mnemonischen Form eingegeben werden können.

Vokabularien dienen ganz allgemein dazu, den definierten Namen einen lokalen Geltungsbereich zu verschaffen.

Forth behandelt die Einträge in die Vokabularien als verkettete Listen, indem bei jedem Namenseintrag eine Adresse (Linkaddress) miteingetragen wird, die auf den letzten zum gleichen Vokabular gehörigen Eintrag zeigt.

Kapitel 3

Vokabular

Im folgenden finden Sie das Stammvokabular von PC-FORTH beschrieben.

Wie in den meisten FORTH-Beschreibungen üblich, sind die Befehlsworte in ASCII-Code-Reihenfolge aufgeführt. Denn da auch beliebige Sonderzeichen als Befehlsworte Verwendung finden, reicht eine alphabetische Reihenfolge nicht aus.

Worte, deren Beschreibung kleingedruckt ist, werden normalerweise nur FORTH-intern verwendet. Meistens handelt es sich dabei um Befehle, die der Compiler anders übersetzt, als Sie sie eingeben (z. B. werden anstelle von IF...ELSE...ENDIF diverse BRANCH-Befehle übersetzt, von denen der Benutzer normalerweise keinen Gebraucht macht). Wenn Sie aber den Compiler erweitern wollen, ist es nützlich, auf diese Interna zurückzugreifen.

In den Beschreibungen wird auf Abweichungen gegenüber den Standard FIG-FORTH ausdrücklich hingewiesen. Alle Befehle aus FIG-FORTH, die nicht die virtuelle Speicherverwaltung von Disketten bzw. Screens betreffen, sind bis auf geringe Ausnahmen in PC-FORTH vollständig kompatibel implementiert. Bei den meisten Befehlen können also auch andere FIG-FORTH-Beschreibungen zu Rate gezogen werden.

Einige Abweichungen gegenüber FIG-FORTH betreffen die Namensgebung: [und] wird grundsätzlich als ((und)) geschrieben, ebenso werden " durch " ersetzt. Statt' verwendet PC-FORTH das Wort TIC.

Über den Befehlsbeschreibungen ist die Wirkung des Befehls auf den Stack angegeben.

Dabei kennzeichnen:

n, nl, n2 vorzeichenbehaftete 16-Bit Zahlen (2 Byte)
Wertebereich: -32768 ... +32767

a, al, a2 vorzeichenfreie 16-Bit-Zahlen (2 Byte), wie sie vorwiegend für Adreßangaben verwendet werden
Wertebereich: 0 ... 65535

»
c

d, dl, d2 vorzeichenbehaftete 32-Bit Zahlen (4 Byte)
Wertebereich: -2147483648... +2147483647

ud, udl, ud2 vorzeichenfreie 32-Bit Zahlen (4 Byte)
Wertebereich: 0 ... 4294967295

Links steht, was sich vor dem jeweiligen Befehl auf dem Stapel befinden muß, rechts was hinterher auf dem Stapel vorgefunden wird.

c

! (n a -)
speichert n bei den Adressen a, a + 1 (das höherwertige Byte von n kommt nach a). Wird in der Form:

n Variablenname !

verwendet, um in einer Variablen den Wert n zu speichern.

!: (n -) oder (a n -)
dient zur Übergabe von Zahlenwerten oder Zeichenfolgen an BASIC-Variable. Kann nur innerhalb von Definitionen verwendet werden, die mit <P eingelesen werden.
Anwendungsform:

n !: "v" Übergabe des Wertes n an die Variable v
a n !: "v$" Übergabe der Zeichenfolge mit der Länge n ab Adesse a an die Textvariable v$

v kann anstelle einer einfachen Variablen auch ein BASIC-Feldelement sein.

Erfolgt bei der Auswertung von v ein Fehler (z. B. nicht dimensioniertes Feld), so kommt die Fehlermeldung ?"basic.

Der Compiler legt für !: die Codeadresse der zugehörigen Laufzeitfunktion (!:) und den Ausdruck v bzw. v$ mit vorangehendem Längenbyte und nachfolgendem Endtrenner ab. Der abgelegte Ausdruck kann dabei auch BASIC-Tokens enthalten.

Nicht in FIG-FORTH enthalten.

(

!CSP (-)

speichert den Stapelpegel (C-Stack) in der Anwendervariablen CSP. (Wird vom Compiler für Strukturprüfungen gebraucht.)

\# (dl -- d2)

bringt die letzte (niederwertigste Stelle) von dl als druckbare ASCII-Ziffer in einen Puffer. Dafür wird eine Division durchgeführt:

d2 = dl/Zahlenbasis (BASE), wobei der Divisionsrest in eine ASCII-Ziffer konvertiert wird. (Verwendung innerhalb der Zahlenausgabe, zwischen <# und #>.)

\# > (d - a n)

beendet Aufbereitung einer Zahl für die Ausgabe:
Löscht d, hinterläßt Anfangsadresse a und Länge n der ASCII-Ziffernfolge so, daß diese mit TYPE ausgegeben werden kann.

\# S (dl -- d2)

bereitet dl als druckbare ASCII-Ziffernfolge auf.
(Wiederholt #, bis d2 = 0.)

\#" (nl ~ n2)

liefert die Nummer des ersten Auftretens eines Zeichens in einer Zeichenfolge. Anzuwenden in der Form:

nl #" Zeichenfolge"

Sucht das Zeichen mit dem ASCII-Code nl in der angegebenen Zeichenfolge. Wird das Zeichen dort gefunden, so ist n2 die Nummer des ersten Auftretens innerhalb der Zeichenfolge, andernfalls hat n2 den Wert 0 (gezählt wird von links nach rechts ab 1). Kann zusammen mit CASE: für Kontrollstrukturen verwendet werden.

Der Compiler speichert anstelle von ¥r die Codeadresse der zugehörigen Laufzeitfunktion (#") und dahinter die Zeichenfolge mit vorangestelltem Längenbyte.

Nicht in FIG-FORTH.

(FIG-FORTH)
identisch TIC im PC-FORTH

&B (al a2 -- a3 a4 n)

Umrechnung von (langen) Hexziffernfolgen in Binärwerte, kann zur Eingabe von Maschinencode verwendet werden:
Eine ab Adresse a2 abgespeicherte Folge von Hexziffern in ASCII-Darstellung wird binär verschlüsselt ab Adresse al abgelegt. Je 2 Ziffern ergeben 1 Byte. Das erste Zeichen, das keine Hexziffer (0 bis 9, A bis F) ist, bricht die Verarbeitung ab. Ein Zwischenraum zwischen Ziffernpaaren bewirkt jedoch keinen Abbruch, sondern wird überlesen. a3 ist die Adresse hinter dem letzten Byte des Ergebnisses; a4 ist die Adresse des Zeichens, das die Verarbeitung abgebrochen hat; n ist dieses Zeichen selbst. Die Länge der umzurechnenden Zeichenfolge ist nur durch den verfügbaren Speicherplatz begrenzt. Sie sollte jedoch geradzahlig sein. Andernfalls wird die letzte Ziffer ignoriert.

Nicht in FIG-FORTH.

((--)

leitet Kommentar ein. Auf die öffnende Klammer muß ein Zwischenraum (SPACE) folgen; dahinter beginnt der Kommentar. Die erste schließende Klammer oder Zeilen-Ende (ENTER) beendet den Kommentar.

(!:) (n --) oder (a n -)

Laufzeitfunktion zu !: wird gefolgt von Längenbyte und BASIC-Ausdruck, dem ein Endtrenner angfügt ist.

(#~) (nl - n2)

Laufzeitfunktion zu #" wird gefolgt von Längenbyte und einer Zeichenfolge, die nach dem Zeichen mit dem ASCII-Code nl abgesucht wird. Auf dem Stack wird n2 hinterlassen, wobei n2 = 0, wenn das Zeichen nicht gefunden wurde, andernfalls hat n2 die Nummer des ersten Auftretens.

Nicht in FIG-FORTH.

(((")

unterbricht innerhalb einer Wortdefinition den Kompilierzustand und schaltet auf direkte Ausführung um. Die folgenden Worte werden nicht kompiliert, sondern sofort ausgeführt, bis mit)) in den Kompilierzustand zurückgeschaltet wird.

In FIG-FORTH [

((COMPILE)) (-)

sorgt innerhalb einer Wortdefinition dafür, daß das nächste IMMEDIATE-Wort kompiliert und nicht ausgeführt wird. IMMEDIATE-Worte wie ; oder ((COMPILE)) werden normalerweise auch innerhalb von Definitionen sofort ausgeführt, vgl. IMMEDIATE.

In FIG-FORTH [COMPILE]

(.") (FIG-FORTH)

identisch (.") in PC-FORTH.

(•") (--)

Laufzeitfunktion zu ." wird gefolgt von Längenbyte und Text. Schreibt den Text in den Ausgabepuffer.

In FIO-FORTH (.")

(;CODE) (-)

Laufzeitfunktion zu ;CODE. Beschreibt das Codefeld des zuletzt definierten Wortes mit einem Zeiger auf den folgenden Maschinencode.

(©:) (--)

Laufzeitfunktion zu ®: wird gefolgt von Längenbyte und BASIC-Ausdruck, dem ein Doppelpunkt angefügt ist.

(+LOOP) (n --)

Laufzeitfunktion zu +LOOP; addiert n zum Schleifenzähler; wird dabei die von (DO) gespeicherte Schranke überschritten oder erreicht, so folgt Schleifen-End-Behandlung. Andernfalls Rücksprung hinter (DO).

(ABORT) (initialisiert Stack)

veranlaßt Warmstart. Wird im Fehlerfall ausgeführt, wenn WARNING den Wert -1 hat.

(DO) (nl n2 --)

Laufzeitfunktion zu DO, schreibt Anfangswert n2 und Schranke nl für den Schleifenzähler in den R-Stapel (R-Stack). Siehe LOOP und +LOOP.

(FIND) (al a2 - a3 nl n2) oder (al a2 -- n3)

sucht einen Namen im Wörterbuch.

al = Namensfeldadresse des zu suchenden Namens im Speicher;
a2 = Namensfeldadresse desjenigen Namens im Wörterbuch, mit dem das Suchen beginnen soll;
a3 = Namensfeldadresse des gefundenen Namens im Wörterbuch;
nl = Längenbyte des gefundenen Namens (einschließlich Immediate-Bit);
n2 = 1 (besagt: "Name gefunden")
n3 = 0 (besagt: "Name nicht gefunden")

(LINE) (**nl** - a n2)
stellt eine BASIC-(Text-)Zeile zur Ausgabe (als Meldung) bereit.

nl = BASIC-Zeilennummer
a = ihre Anfangsadresse (hinter dem Längenbyte)
n2 = ihr Längenbyte

Nach (LINE) kann die Zeile mittels TYPE in den Ausgabepuffer geschrieben werden.

In FIG-FORTH mit anderer Bedeutung.

(LOOP) (-)
Laufzeitfunktion zu LOOP erhöht den Schleifenzähler auf dem R-Stack um 1; wird dabei die von (DO) gespeicherte Schranke überschritten oder erreicht, so folgt die Schleifen-End-Behandlung.
Andernfalls Rücksprung hinter (DO) (siehe LOOP).

(NUMBER) (dl al - d2 a2)
konvertiert die ASCII-Ziffernfolge ab al + 1 akkumulierend zu dl. Das Resultat ist d2.

a2 ist die Adresse des ersten Zeichens, das nicht mehr als Ziffer in dem durch BASE bestimmten Zahlensystem konvertierbar ist.

)) (--)
schaltet innerhalb einer Wortdefinition in den Kompilierzustand zurück, nachdem dieser mit ((ausgeschaltet wurde, vgl. ((.

In FIG-FORTH 1

(nl n2 -- n3)
n3 = n2 * nl

*/ (nl n2 n3 -- n4)
n4 = nl * n2/n3

Die Berechnung erfolgt im 32-Bit-Zahlenbereich, ist also noch verwendbar, wenn die Wortfolge nl n2 * n3 / versagen würde.

*/MOD (nl n2 n3 -- n4 n5)
n4 ist der Rest, n5 der Quotient der Division (nl * n2)/n3.

Die Berechnung erfolgt im 32-Bit-Zahlenbereich, ist also noch anwendbar, wenn die entsprechende 16-Bit-Rechnung versagen würde.

\+ (nl n2 - n3)
n3 = nl + n2

+! (n a -)
zu der bei den Adressen a, a+ 1 gspeicherten 16-Bit-Zahl wird n addiert.

+- (nl n2~ n3)
n3 = nl, wenn n2 > =0; n3 = -nl, wenn n2 <0.

+LOOP(n -)
markiert das Ende einer Zählschleife, die mit DO eröffnet wurde. Anzuwenden in der Form:

nl n2 DO Wortfolge n3 + LOOP

nl = Schranke, n2 = Anfangswert, n3 = Schrittweite.
Die Schleife wird sooft durchlaufen, bis der Schleifenzähler die Schranke nl erreicht oder überschreitet, jedoch mindestens ein Mal. Die Zahlen nl, n2, n3 müssen nicht explizit angegeben werden, sondern können auch als berechnete Ergebnisse im Stapel (Stack) stehen.

Arbeitsweise:
Der Compiler speichert anstelle von DO bzw. + LOOP die Codeadresse der zugehörigen Laufzeitfunktion (DO) bzw. (+ LOOP).

Wird später die kompilierte Funktion durchlaufen, so arbeiten (DO) und (+ LOOP) wie folgt:

(DO) schreibt nl als Schranke und n2 als Anfangswert des Schleifenzählers in den R-Stapel; dann wird die Wortfolge hinter (DO) durchlaufen. (+ LOOP) addiert die Schrittweite n3 zu dem augenblicklichen Wert des Schleifenzählers im R-Stapel.

Wird dabei die Schranke nl überschritten, (bzw. bei negativer Schrittweite unterschritten) oder erreicht, so erfolgt Schleifen-End-Behandlung: nl und n2 werden aus dem R-Stapel gelöscht, und das Programm wird hinter (+ LOOP) fortgesetzt. Andernfalls erfolgt ein Rücksprung auf den Anfang der hinter (DO) gespeicherten Wortfolge.

+ **ORIGIN** (n - a)
a = n + ORIGIN
(ORIGIN) ist die Adresse, ab der das FORTH-System abgespeichert ist. Dort stehen hinter den Sprungbefehlen auf Kalt-, Warm- und Heißstart die Anfangswerte für die Benutzervariablen und andere Systemkonstanten. Für deren Adressierung wird + ORIGIN gebraucht (vgl. Anhang B).

(n -)
speichert n ab der nächsten freien Dictionary-Adresse und erhöht den Dictionary-Zeiger DP um 2.

(nl n2 - n3)
n3 = nl - n2

-DUP (n - n n) oder (n - n)
dupliziert die oberste Stapelzahl, wenn sie nicht Null ist.
Mit -DUP kann z. B. die verhältnismäßig häufige Abfrage

Wert DUP IF Wortfolge ELSE DROP ENDIF

wie folgt vereinfacht werden:

Wert -DUP IF Wortfolge ENDIF

-FIND (- a nl n2) oder (- n3)
bringt aus der Eingabe-Zeichenfolge das nächste (mit Zwischenraum abgeschlossene) Wort an die Stelle HERE und sucht es im CONTEXT- und CURRENT-Wörterbuch.

Wenn gefunden, wird seine Parameterfeldadresse a, sein Längenbyte (einschließlich Immediate-Bit) nl und das Kennzeichen n2= 1 (gefunden) gestapelt; andernfalls nur das Kennzeichen n3 = 0 (nicht gefunden).

-TRAILING (a nl -- a n2)

a = Anfangsadresse einer (mit TYPE auszugebenden) Zeichenfolge,
nl = ihre Länge,
n2 = Länge der Zeichenfolge nach Streichung der rechtsstehenden Blanks.

(n -)
Ausgabe von n in dem mit BASE eingestellten Zahlensystem. Anschließend wird ein Leerzeichen (SPACE) ausgegeben.

(FIG-FORTH)
identisch mit ." in PC-FORTH

.CPU (--)
bringt den Namen LH5801 des im PC-1500 verwendeten Mikroprozessors in die Anzeige.

.LINE (n -)
bringt die BASIC-Text-Zeile mit der Nummer n in die Anzeige. (Wenn nicht vorhanden, kommt Fehlermeldung ~ Zeile n).

In FIG-FORTH andere Bedeutung.

.R (nl n2 --)
gibt nl in einem Feld der Länge n2 rechtsbündig aus.

.S (-)
gibt alle Einträge des C-Stapels als Zahlen aus, ohne den Stapelinhalt zu ändern. (Berücksichtigt BASE.)

(--)
Ausgabe eines Textes. Anzuwenden in der Form:

." Text"

Bringt den angegebenen Text zur Ausgabe. Das Ende des Textes wird mit" oder durch das Zeilenende markiert.

Der Compiler speichert die Codeadresse der zugehörigen Laufzeitfunktion (.") und dahinter den angegebenen Text mit vorangestelltem Längenbyte.

In FIG-FORTH ."

/ (nl n2 - n3)
n3 = nl/n2

/MOD (nl n2 -- n3 n4)
n3 ist der Rest, n4 der Quotient der Division nl/n2.

0 1 2 3
Diese kleinen Zahlen wurden als Konstante in das Wörterbuch mit aufgenommen, da sie sehr häufig gebraucht werden. Auf diese Weise werden jeweils nur 2 Byte im kompilierten Programm benötigt und nicht 4, um diese Konstanten darzustellen.

0< (nl - n2)
n2 = 1, wenn nl < 0, sonst n2 = 0.

0= (nl - n2)
n2 = 1, wenn nl = 0, sonst n2 = 0.
Unter anderem zur Negation einer Bedingung anwendbar, daher auch manchmal mit NOT bezeichnet.

OBRANCH (n -)
Laufzeitfunktion zu IF, UNTIL und WHILE. Hinter der Codeadresse von OBRANCH speichert der Compiler die relative Sprungadresse. Beim Ablauf wird ein Sprung ausgeführt, wenn n = 0 ist. Sonst wird das Programm hinter der relativen Sprungadresse fortgesetzt.

1 + (nl -- n2)
n2 = nl + 1

2! (d a - -)
speichert den 32-Bit Wert d ab Adresse a (4 Bytes).

2 + (nl - n2)
n2 = nl + 2

2@ (a -- d)
lädt einen 32-Bit Wert d von Adresse a auf den Stack.

2DUP (d -- d d)
dupliziert den obersten d 32-Bit Wert auf dem Stapel. Kann natürlich auch für (nl n2 — nl n2 nl n2) verwendet werden.

leitet eine Wortdefinition ein. Anwendung in der Form:

: Name Wortfolge ;

Im Wörterbuch wird unter dem angegebenen Namen die kompilierte Wortfolge gespeichert. Wenn der Name später aufgerufen wird, so wird die kompilierte Wortfolge ausgeführt.

Der Compiler erzeugt im Dictionary einen Eintrag folgender Art:

```
  l      Name     link     de    kompilierte Wortfolge    se
  ↑                 ↑       ↑     ↑
 NFA               LFA     CFA   PFA
```

Längenbyte 1:

Bit	7	6	5	4	3	2	1	0
Bedeutung	1	im	sm	Namenslänge				

Name wie angegeben, doch wird sein letztes Zeichen dadurch gekennzeichnet, daß das höchstwertige Bit= 1 gesetzt wird.

link = Anfangsadresse des letzten zuvor definierten Dictionary-Eintrags
de = Startadresse der zum Doppelpunkt gehörigen Laufzeitfunktion
se = Codeadresse der zu Semicolon gehörigen Laufzeitfunktion ;S
NFA = Namensfeldadresse
LFA = Linkfeldadresse
CFA = Codefeldadresse
PFA = Parameterfeldadresse

Der Doppelpunkt baut diesen Eintrag bis de auf.

Im Längenbyte setzt er das Bit 7 auf 1, um das Längenbyte als solches zu kennzeichnen, sowie das Bit 5, um die Definition als "noch offen" zu charakterisieren. Bit 6 kann durch das Wort IMMEDIATE nach Abschluß der Definition gesetzt werden; Bit 5 wird durch das Semicolon mittels SMUDGE zurückgesetzt. Zuletzt setzt der Doppelpunkt die Anwendervariable STATE auf Hex CO, damit die nun folgenden Worte der "Wortfolge" nicht ausgeführt sondern kompiliert werden. (Ausnahmen bilden jedoch die als "immediate" gekennzeichneten Worte: Diese werden auch im Kompilierzustand ausgeführt.)

Die zum Doppelpunkt gehörige Laufzeitfunktion bewirkt, daß der augenblickliche Stand des FORTH-Befehlszeigers im R-Stapel gestapelt und der Befehlszeiger auf den Anfang der kompilierten Wortfolge gesetzt wird. ;S führt den Rücksprung zur aufrufenden Funktion durch, indem die oberste Adresse im R-Stapel gelöscht und wieder in den FORTH-Befehlszeiger gespeichert wird.

beendet eine Wortdefinition (siehe Beschreibung zum :).

;CASE

wird zusammen mit CASE: gebraucht. Beschreibung siehe CASE:.
Nicht in FIG-FORTH.

;S

Laufzeitfunktion zum Semicolon (siehe Beschreibung zum :).

< (nl n2 -- n3)
n3 = 1, wenn nl < n2; sonst n3 = 0

< #

startet Aufbereitung einer 32-Bit-Zahl für die Ausgabe.
Diese Funktion wird zusammen mit # #S SIGN #> benutzt.

<BUILDS

Wird zusammen mit DOES> zur Erzeugung neuer Datentypen (wie etwa String, Array usw.) verwendet.
Anwendungsform:

: Typname <BUILDS Wortfolgel DOES> Wortfolge2 ;

Typname ist ein Name, der den neuen Datentyp bezeichnet;
Wortfolge 1 wird durchlaufen, wenn ein Objekt des neuen Datentyps definiert wird;
Wortfolge 2 wird durchlaufen, wenn der Name eines solchen Objekts aufgerufen wird.

<R> (n a -)

n wird bei Adresse a, a + 1 gespeichert; zuvor werden im R-Stapel Informationen abgelegt, die bewirken, daß der alte Inhalt von a, a + 1 wiederhergestellt wird, sobald ein Rücksprung von der laufenden Funktion erfolgt.

Nicht in FIG-FORTH.

<P (n -)

bewirkt, daß die FORTH-Eingabe nicht von der Tastatur erfolgt, sondern aus dem BASIC-Programmspeicher ab Zeile mit der Nummer n. PC-FORTH liest Zeile für Zeile, wobei das Anführungszeichen am Beginn jeder Zeile ignoriert wird. Sobald P> erscheint, wird die Eingabe wieder auf die Tastatur umgeschaltet.

Wird beim Übersetzen aus den BASIC-Zeilen ein Fehler festgestellt, so wird der FORTH-Fehlermeldung ein #-Zeichen und die Nummer der fehlerhaften Zeile angefügt; außerdem wird der Tastatur-Eingabemodus wieder eingeschaltet. Die fehlerhafte Definition bleibt ggf. offen und kann von Hand zu Ende geführt oder mit ((SMUDGE FORGET Name gelöscht werden.

Wird mit n eine nicht existierende Programmzeile angegeben, so kommt die Fehlermeldung "Zeile ~ def #n".

Ist einer Fehlermeldung eine negative Nummer angefügt, so hat der Anwender das abschließende P> vergessen.

(nl n2 - n3)
n3 = 1, wenn nl = n2; sonst n3 = 0

> (nl n2 -- n3)
n3 = 1, wenn nl > n2; sonst n3 = 0

>R (n -)
n wird im C-Stapel gelöscht und auf den R-Stapel gelegt.

? (a -)
Die bei den Adressen a, a + 1 gespeicherte Zahl n wird mittels . ausgegeben. Abkürzung von © .

?COMP
gibt die Fehlermeldung "nur in :Def" aus, wenn nicht der Kompilierzustand eingeschaltet ist.

?CSP
gibt die Fehlermeldung "Strukturfehler" aus, wenn der Stapelpegel sich gegenüber dem in CSP sichergestellten Stapelpegel geändert hat. (Wird vom Compiler benutzt, um sicherzustellen, daß Wortgruppen wie IF ELSE ENDIF nur in strukturell richtiger Weise kompiliert werden können.)

7ERROR (nl n2 -)
Die Fehlermeldung Nummer n2 (BASIC-Zeile OFFSET + n2) wird ausgegeben, wenn nl nicht = 0 ist.

7EXEC
gibt Fehlermeldung "~ kompilierbar" aus, wenn nicht der Ausführungszustand eingeschaltet ist.

?LOADING
gibt Fehlermeldung "~ ladend" aus, wenn nicht die Eingabe aus BASIC-Zeilen "geladen" wird. Nicht in FIG-FORTH.

?OI (- n)
n = 1, wenn der Anwender den letzten FORTH-Ausgabestop mit • quittiert hat; andernfalls n = 0. Nicht in FIG-FORTH.

(

?PAIRS (nl n2 -)
gibt Fehlermeldung "passt nicht" aus, wenn nl *i^nl* ist. (Wird vom Compiler benutzt, um sicherzustellen, daß verschiedene Wortgruppen wie IF ELSE ENDIF nicht in unpassender Weise gemischt werden.)

7STACK
gibt eine Fehlermeldung "leer" oder "voll" aus, wenn die Stapelgrenzen überschritten werden.

?TERMINAL (- n)
n = 1, wenn während der Ausführung von "reinem" FORTH-Code die BREAK-Taste gedrückt wurde; sonst 0.
Bei der FORTH EinVAusgabe über BASIC führt die BREAK-Taste zur Programmunterbrechung.

© (a ~ n)
lädt die bei den Adressen a, a + 1 gespeicherte 16-Bit Zahl n auf den Stack. Wird in der Form:

Variblenname ©

verwendet, um den Inhalt einer Variablen auf den Stapel zu laden.

©: (— al) oder (— al n)
dient zur Übernahme von Zahlen und Zeichenfolgen aus BASIC in FORTH. Anwendung in Wortdefinitionen, die mittels <P eingelesen werden.
Anwendungsform:

©: "b"

wobei b ein beliebiger BASIC-Ausdruck ist (z. B. der Name einer BASIC-Variablen), der kein Anführungszeichen enthält.

Wenn der angegebene BASIC-Ausdruck b ein numerisches Resultat al hat, wird dieses auf den Stapel gelegt (— al).

Wenn das Resultat des angegebenen BASIC-Ausdrucks b eine Zeichenfolge ist, so wird ihre Anfangsadresse a2 und ihre Länge n auf den Stapel gelegt (-- a2 n).

Ein numerischer Wert wird nur dann übernommen, wenn er im Bereich 0 bis 65535 liegt, wobei Stellen nach dem Dezimalpunkt abgeschnitten werden. Die Berechnung von b wird jedoch im vollen BASIC-Zahlenbereich ausgeführt.

Falls b ein nichtnumerisches Resultat hat, so wird die Zeichenfolge in dem Bereich gespeichert, den BASIC für diesen Zweck verwendet. Sie wird deshalb überschrieben, sobald BASIC eine andere Zeichenfolge verarbeitet (z. B. bei Ausgabe von "OK"). Falls sie länger gebraucht wird, muß sie sichergestellt werden (z. B. durch CMOVE in den FORTH-Speicher oder duch Zuweisung an eine BASIC-Variable mittels !:).
Nicht in FIG-FORTH.

Erkennt BASIC bei der Verarbeitung von b einen Fehler, so gibt FORTH die Fehlermeldung ?"basic aus.

Der Compiler legt für ®: die Codeadresse der zugehörigen Laufzeitfunktion (®:) und den (basicverschlüsselten) Ausdruck b mit vorangehendem Längenbyte und nachfolgendem Endetrenner ab.

ABORT (Löscht Stack)
führt Warmstart aus: Beide Stapel werden gelöscht, der Ausführungszustand wird eingeschaltet, BASE erhält den Wert 10, die Meldung PC-FORTH und die Versionsbezeichnung werden ausgegeben.

ABS (nl ~ n2)
n2 = Absolutbetrag von nl

AGAIN

zu verwenden in der Form:

BEGIN Wortfolge AGAIN

bewirkt unbedingte Wiederholung der Wortfolge.
Diese Schleife kann nicht mehr verlassen werden, nicht einmal mit BREAK — es sei denn, sie ruft irgendeine Funktion auf, die nicht unbedingt in die Schleife zurückkehrt. (Beispiel: CR, auch wenn es implizit erfolgt. Man kann dann BREAK und Kalt- oder Warmstart geben.)

Der Compiler übersetzt AGAIN in einen unbedingten Sprung auf die Stelle hinter BEGIN.

AIN (- a)
liefert die Adresse des nächsten Zeichens aus der Eingabezeichenfolge. Berücksichtigt BLK.

ALLOT (n -)
addiert n zum Dictionary-Zeiger DP (HERE). Damit kann im Dictionary Speicherplatz reserviert oder (da n auch <0 sein darf) nachträglich bespeichert werden.

AND (nl n2 -- n3)
nl und n2 werden bitweise durch logisches "Und" verknüpft. Ein Bit von n3 wird genau dann auf 1 gesetzt, wenn das entsprechende Bit in nl **und** das entsprechende Bit in n2 gesetzt (= 1) ist.

BAL (n -)
verzweigt ins BASIC zur Programmzeile n (1 ä n £ 255). Das dort beginnende BASIC-Programm wird ausgeführt. Der Rücksprung ins FORTH erfolgt mit

GOTO "H"

wobei das FORTH-Programm an der Unterbrechungsstelle fortgesetzt wird.

Dabei dürfen die von FORTH verwendeten BASIC-Variablen OS, I$(0) und O$(0) nicht verändert werden.

Existiert die angegebene Zeile nicht, so erfolgt im BASIC die Fehlermeldung ERROR 11.

Nicht in FIG-FORTH.

BACK (a -)
speichert HERE-a als 16-Bit-Zahl ab HERE. Wird vom Compiler benutzt, um relative Rückwärtssprungadressen zu speichern. (Die absolute Sprungzieladresse a wird vorher beim Kompilieren des Wortes BEGIN oder DO auf den Stapel gelegt.)

B# (n -- a)
liefert die Anfangsadresse a der BASIC-Zeile mit der Nummer n. Es wird die Adresse des ersten Zeichens dieser Zeile nach Zeilennummer und Längenbyte übergeben. Falls keine Zeile mit der Nummer n existiert ist a = 0.

Nicht in FIG-FORTH.

BASE (- a)

Die Anwendervariable BASE enthält als Wert die für Ein- und Ausgabekonversionen gültige Zahlenbasis.

Nach Kalt- und Warmstart enthält BASE den Wert 10, so daß die Ein- und Ausgabe von Zahlen dezimal erfolgt.

BEGIN (-)

Beschreibung siehe **UNTIL,** WHILE, AGAIN

Der Compiler stapelt die Adresse HERE für die Berechnung der relativen Rückwärtssprungadresse, sowie eine Kennzeichnung für die Strukturprüfung.

BL (- n)

Die Konstante BL stapelt n = 32, also den ASCII-Wert des Leerzeichens (BLANK, SPACE).

BLANKS (a n -)

speichert n Leerzeichen (SPACE) ab Adresse a.

BLK (-a)

Die Anwendervariable BLK enthält als Wert die aktuelle Zeilennummer, während die FORTH-Eingabe aus Zeilen des Programmspeichers erfolgt.

n <P speichert die Zeilennummer n bei der Adresse a der Anwendervariablen BLK. Ist eine Zeile abgearbeitet, so kommt die Nummer der nächsten Zeile nach a. Endet eine Zeile mit P>, so wird die 16-Bit-Zahl 0 bei a gespeichert. Diese 0 ist zugleich das Kennzeichen für den manuellen Eingabemodus.

In FIG-FORTH andere Bedeutung.

BR (- a)

Die Anwendervariable BR enthält als Wert den CPU-Stapelpegel (also den Wert des S-Registers) vor dem Aufruf des FORTH-Systems. Dieser Wert wird bei jedem Rücksprung aus dem FORTH-System wieder in das S-Register gebracht.

Nicht in FIG-FORTH.

BRANCH

bewirkt unbedingten Sprung. Der Compiler speichert beim Verarbeiten von ELSE, AGAIN und REPEAT die Codeadresse von BRANCH und dahinter die relative Sprungzieladresse.

C! (n a -)

speichert n (0£n<256) bei der Adresse a. (Dabei werden nur die niederwertigen 8 Bits von n auf Adresse a gespeichert, der Inhalt von a + 1 wird nicht verändert.)

C# (nl - n2)

konvertiert einen Ziffernwert nl in den zugehörige ASCII-Code n2. BASE wird nicht berücksichtigt. Also:

n2 = nl + 48, wenn 0 ^ nl S 9
n2 = nl + 55, wenn 10 S nl 2 200

Nicht in FIG-FORTH.

C, (n -)

Die niederwertigen 8 Bits von n werden auf die Adresse HERE gespeichert. C, speichert also 1 Byte im Dictionary, HERE wird um 1 erhöht.

C© (a -- n)

lädt den Inhalt der Speicherzelle mit der Adresse a auf den Stack, n ist der Wert des dort gespeicherten Bytes (0 2 n < 256).

CASE:

dient zur Definition von "Verteilerfunktionen" und ist in folgender Weise anzuwenden:

CASE: ex cO cl c2 .. cn ;CASE

Dabei müssen cO bis cn die Namen vorher definierter FORTH-Worte sein; als ex ist der gewünschte Name der zu definierenden Verteilerfunktion anzugeben. (Dieser darf neu sein.)

Nachdem diese Verteilerfunktion ex so definiert ist, kann sie manuell oder in Wortfolgedefinitionen aufgerufen werden.

Soll die Funktion ex ausgführt werden, so muß zuoberst im Stapel eine Nummer nl stehen, die einen der Werte 0 bis n hat und angibt, welche der Funktionen cO bis cn aufgerufen werden soll.

Die Tätigkeit der Funktion ex besteht darin, die Nummer nl löschend aus dem Stapel zu holen und dann die nl-te Funktion cnl (als einzige) aufzurufen. Nach dem Rücksprung von cnl wird die Verarbeitung hinter ;CASE fortgesetzt.

Nicht in FIG-FORTH.

CFA (al -- a2)
CFA rechnet die Parameterfeldadresse al in die zugehörige Codefeldadresse a2 um. In PC-FORTH ist a2 = al-2.

CLS (-)
löscht den Ausgabepuffer und setzt den Ausgabezeiger OUT auf 0.

Nicht in FIG-FORTH.

CMOVE (al a2 n -)
kopiert n Bytes ab Adresse al nach a2.

Für den Fall überlappender Bereiche ist von Bedeutung, daß zuerst Inhalt von al nach a2 kopiert wird, dann Inhalt von al +1 nach a2+ 1 usw.

(Sollte diese Arbeitsweise unerwünscht sein, verwendet man CX)

COLD (initialisiert Stack)
bewirkt Kaltstart. Löscht alle Anwenderdefinitionen, speichert Anfangswerte in die Anwendervariablen und führt dann Warmstart (ABORT) aus.

COMPILE
Wird im Gegensatz zu ((COMPILE)) vom Compiler ganz normal übersetzt und so gespeichert und hat auf die augenblickliche Tätigkeit des Compilers keinerlei Wirkung.

Wird jedoch später die Funktion, in deren Definition COMPILE auftrat, aufgerufen, so wird das hinter COMPILE stehende übersetzte Wort nicht ausgeführt, sondern im Dictionary ab HERE gespeichert.

(

CONSTANT (n --)

erzeugt eine Konstante mit dem Wert n. Anzuwenden in der Form:

n CONSTANT name

Bei späterem Aufruf des Namens wird der Wert n auf den Stapel gelegt.

CONTEXT (- a)

Der Wert dieser Benutzervariable verweist auf das Wörterbuch, in dem die eingegebenen Worte zuerst gesucht werden sollen.

(**COUNT** (al -- a2 n)

COUNT erwartet als obersten Stapeleintrag die Anfangsadresse al eines Text-Eintrages, der mit einem Längenbyte beginnt; a2 = al + 1 ist die Anfangsadresse des eigentlichen Textes; das niederwertige Byte von n ist der Wert des Längenbytes, das höherwertige Byte von n ist = 0. (Die beiden Werte a2 und n werden so von TYPE im Stapel erwartet.)

CR (-)

bewirkt die tatsächliche Ausgabe der Zeichen, die (von EMIT, TYPE, ., ." usw.) in den Ausgabepuffer gebracht wurden.

Die Ausgabe geschieht im BASIC in der Zeile 38. Dabei wird der Drucker mit der Basicvariable L gesteuert.

Wenn der Drucker eingeschaltet ist (L = 1), erfolgt die Ausgabe auf Drucker und Display. Anschließend wird, falls keine Taste betätigt wird, die Programmausführung fortgesetzt, anderfalls erfolgt ein Ausgabestop.

Ist der Drucker abgeschaltet (L = 0), so erfolgt die Ausgabe nur auf der Anzeige. Anschließend wird ein Ausgabestop durchgeführt und auf die Betätigung einer Taste gewartet, worauf dann die Programmausführung fortgesetzt wird. Wird dabei die Taste i betätigt, so wird der TRACE eingeschaltet, mit der Taste T kann der TRACE wieder ausgeschaltet werden.

CREATE

wird angewendet in der Form:

CREATE Name

um einen Namenseintrag im Wörterbuch zu erzeugen. Der Namenseintrag hat folgende Form:

1 Name link cfa

Das Längenbyte 1 enthält in den unteren 5 Bits (Bit 0 ... Bit 4) die Länge des Namens, Bit 5 und Bit 7 werden auf 1 gesetzt. Der Name wird gespeichert, wie angegeben; nur wird sein letztes Zeichen durch eine 1 in Bit 7 gekennzeichnet, link ist die Anfangsadresse des jüngsten vorher definierten Namenseintrages, cfa ist die Adresse des darauffolgenden Bytes.

CSP (- a)
Die Anwendervariable CSP wird vom Compiler für Strukturprüfungen gebraucht. Ihr Wert ist der augenblickliche Stapelpegel ("current Stack Position") nach Eröffnung einer Wortdefinition. Er muß mit dem Stapelpegel beim Abschließen der Definition (mittels Semicolon) übereinstimmen; sonst "Strukturfehler".

CSWAP (nl ~ n2)
vertauscht das höherwertige Byte von nl mit dem niederwertigen Byte, also die obersten beiden Bytes im Stapel.

Nicht in FIG-FORTH.

CURRENT (- a)
Der Inhalt der Anwendervariablen CURRENT zeigt auf das Vokabulpr, in das die augenblicklichen Definitionen eingetragen werden.

Am Anfang ist dies das FORTH-Vokabular. Mit DEFINITIONS kann jedoch das CONTEXT-Vokabular für den Eintrag von Definitionen aktiviert werden.

CX (al a2 n -)
vertauscht den Speicherbereich ab Adresse al mit dem Speicherbereich ab a2 in der Länge von n Bytes. Für den Fall überlappender Bereiche ist von Bedeutung, daß zuerst der Inhalt des letzten Bytes im ersten Bereich mit dem Inhalt des letzten Bytes im zweiten Bereich vertauscht wird, dann die Inhalte der vorletzten Bytes usw. Die Verarbeitung erfolgt also in umgekehrter Richtung gegenüber CMOVE.

Nicht in FIG-FORTH.

D+ (d l d 2 - d 3)
d3 = dl + d2

D + - (d l n - - d 2)
d2 = dl, wenn n > 0 oder n = 0; sonst d2 = -dl

D. (d -)
gibt eine doppeltgenaue Zahl (32-Bit) aus. Dabei werden die höherwertigen 2 Bytes an oberster Stelle im Stapel erwartet. Um die Doppelzahl Hex 11223344 zu erhalten, kann man also HEX 3344 1122 D. eingeben.

D.R (d n -)
gibt eine doppeltgenaue Zahl d rechtsbündig in einem Feld der Länge n aus.

DABS (dl -- d2)
d2 = Absolutbetrag von dl

DEC (-)
setzt den Wert von BASE auf 10, so daß Ein- und Ausgabe von Zahlen dezimal erfolgen. Heißt in FIG-FORTH DECIMAL.

DECIMAL (FIG-FORTH)
Heißt im PC-FORTH einfach DEC

(

DEFINITIONS
setzt Wert von CURRENT auf den Wert von CONTEXT.

Damit werden neue Definitionen in das CONTEXT-Vokabular aufgenommen, in dem der Compiler (mittels -FIND) die zu überseztenden Namen (zuerst) sucht.

DIGIT (nl n2 - n3 n4, wenn durchführbar)
(nl n2 — n4, wenn nicht durchführbar)
Konvertierung einer Ziffer im ASCII-Code in den zugehörigen Zahlenwert.

\

nl = ASCII-Wert des Zeichens
n2 = Basis des zu verwendenden Zahlensystems
n3 = resultierender Wert
n4 = 1, wenn Konversion durchführbar; n4 = 0, wenn Konversion nicht durchführbar

DLITERAL (d -)
übersetzt während des Kompilierens eine zuvor gestapelte doppeltlange Zahl so, daß beim Durchlaufen des erzeugten FORTH-Programmcodes diese Doppelzahl auf den Stapel gelegt wird. Das erzeugte Programmstück lautet im Prinzip:

LIT untere Hälfte LIT obere Hälfte

wobei natürlich anstelle von LIT die Codeadresse von LIT gespeichert wird und die beiden Hälften der Doppelzahl binär verschlüsselt sind (vgl. LITERAL).

DMINUS (dl - d2)
d2 = -dl (Darstellung im Zweierkomplement)

DO (nl n2 --)
eröffnet eine Zählschleife mit dem Anfangswert n2 und der Schranke nl. Wird in den beiden Wortgruppen

DO .. LOOPund
DO .. +LOOP

verwendet.

Beschreibung siehe LOOP bzw. + LOOP.

DOES> (a -)
wird in den Wortgruppen

<BUILDS .. DOES> ..

verwendet.

Beschreibung siehe <BUILDS.

DP (- a)
Der Wert dieser Anwendervariablen ist der Dictionary-Zeiger, sie enthält die Adresse des ersten freien Bytes im Dictionary. Der Wert von DP kann mit HERE auf den Stapel gelegt und mit ALLOT geändert werden.

DPL (- a)
Der Wert dieser Anwendervariablen wird beim Konvertieren einer als ASCII-Ziffernfolge vorliegenden Zahl gesetzt, wie es z. B. bei der Zahleneingabe erfolgt. Sie enthält die Anzahl der Ziffern nach dem Dezimalpunkt. Wird kein Dezimalpunkt gefunden, so wird die Zahl als einfachgenauer Wert konvertiert und DPL = -1 gesetzt. Wird ein Dezimalpunkt gefunden, so entsteht ein doppeltgenauer Wert, und DPL erhält einen Wert > 0.

FORTH arbeitet intern nur mit ganzen Zahlen. Mit Hilfe von DPL kann der Anwender jedoch selbst Festkommazahlen behandeln, wie es z. B. bei Geldbeträgen in DM sinnvoll ist.

DROP (n -)
Die oberste Zahl im Stapel wird gelöscht.

DUMP (a n -)
gibt einen Speicherbereich von n Bytes Länge ab Adresse a mittels . aus.

DUP (n - n n)
dupliziert die oberste Zahl im Stapel.

ELSE (-)
wird in der Wortfolge

.. IF .. ELSE .. ENDIF gebraucht. Beschreibung siehe IF.

EMIT (n -)
schreibt ein Zeichen in den Ausgabepuffer. Dabei wird das niederwertige Byte von n als ASCII-Wert des auszugebenden Zeichens interpretiert. Der Ausgabezeiger OUT wird um 1 erhöht. Falls der Wert von OUT 24 erreicht, fügt EMIT das Zeichen ~ als Ausgabetrenner an und gibt den Ausgabepuffer aus.

ENCLOSE (a nl - a n2 n3 n4)

ENCLOSE heißt "einschließen". Diese Funktion wird z. B. von der Wortlesefunktion WORD benutzt, um in der Eingabezeichenfolge das nächste FORTH-Wort zu ermitteln und in Zeiger "einzuschließen" (d. h. Zeiger auf seinen Anfang und sein Ende zu setzen).

Mittels ENCLOSE werden aber auch Zahlen, Kommentare, Texte und in FORTH eingefügte BASIC-Ausdrücke "eingeschlossen".

a = Adresse, ab der die ENCLOSE-Funktion suchen soll
nl = Trennzeichen (ASCII-verschlüsselt), z. B. 32 = Blank, 41 = ")" usw.
n2 = Abstand von a bis zum ersten Zeichen des gefundenen Wortes (im allgemeinen nach Überlesen voranstehender Blanks bzw. Trenner)
n3 = Abstand von a bis zu dem 1. Zeichen hinter dem gefundenen Wort
n4 = Abstand von a bis zu der Adresse, ab der die ENCLOSE-Funktion beim nächsten Aufruf suchen soll.

Wie im Original FIG-FORTH betrachtet ENCLOSE auch eine binäre Null als (außergewöhnlichen) Trenner; im Gegensatz zum FIG-FORTH gibt es noch einen weiteren außergewöhnlichen Trenner, nämlich das BASIC-Zeilen-Endzeichen (ENTER), dessen ASCII-Code 13 ist. Dieses muß dafür sorgen, daß bei Eingabe von FORTH-Quellcode aus BASIC-Zeilen die Zeilennummer nicht mitübersetzt, jedoch (für etwa notwendige Fehlermeldungen) in BLK aufbewahrt wird.

END (n -)

gleichbedeutend mit UNTIL

ENDIF (--)

wird in den Wortgruppen .. IF .. ELSE .. ENDIF und .. IF .. ENDIF gebraucht und bei IF beschrieben.

ERASE (a n - -)

löscht einen Speicherbereich von n Bytes Länge ab Adresse a mit binären Nullen.

ERROR (n -)

dient zur Ausgabe von Fehlermeldungen.
Falls WARNING den Wert -1 hat, wird jedoch die Meldung unterdrückt und ein Warmstart (ABORT) mit Löschen beider Stapel veranlaßt. Der Anwender kann (mit entsprechender Vorsicht) statt des Warmstarts eine eigene Fehlerbehandlungsfunktion zum Aufruf bereitstellen.

Ist der Wert von WARNING = 0, so wird der Name der Funktion, in welcher der Fehler erkannt wurde, sowie die Fehlernummer n (mit vorangestelltem Fragezeichen und Anführungszeichen) ausgegeben. Hat WARNING den Wert 1, so wird statt der Fehlernummer die BASIC-Zeile mit der Nummer Offset + n als Fehlertext ausgegeben (wobei Offset der Wert der Anwendervariablen OFFSET ist).

Sollte diese auszugebende Zeile nicht definiert sein, so kommt statt des Textes die Meldung "~ Zeile" und die Nummer der nicht definierten Zeile; der Anwender kann durch Subtraktion des Werts von OFFSET die Fehlernummer ausrechnen.

EXECUTE (a -)

führt die Funktion aus, deren Codefeldadresse a zuoberst im Stapel steht.

EXPECT (a n -)

holt bis zu n beliebige Zeichen von der Tastatur, speichert sie ab Adresse a und gibt sie zugleich aus. Beendet wird EXPECT entweder bei Eingabe des n+1-ten Zeichens oder vorzeitig bei Eingabe eines Steuerzeichens. (Ein beendendes Steuerzeichen ist ein Zeichen mit ASCII-Code<32, ausgenommen die Umschalt-Zeichen DEF, SHIFT, SML.)

In jedem Falle wird den abgespeicherten Zeichen eine binäre Null angefügt (so daß der freizuhaltende Bereich um 1 größer sein muß als n); und das beendende Zeichen wird als Wert der Anwendervariable OIV gespeichert, wo es für vielleicht erwünschte Ablaufsteuerungen verfügbar ist.

(Im FIG-FORTH ist ENTER das einzige Zeichen, das die EXPECT-Eingabe vorzeitig beenden kann; die Wirkung der anderen Steuerzeichen ist nicht ausdrücklich festgelegt. Ferner ist offen, wie viele binäre Nullen als Endbegrenzer angefügt werden.)

FENCE (~ a)

Der Wert der Anwendervariable FENCE ("Zaun") wirkt als untere Schranke für die FORGET-Funktion.

Worte, deren Namenseinträge unterhalb der Adresse a beginnen, können nicht (bzw. nur nach Änderung des Wertes von FENCE) mit FORGET gelöscht werden. Kaltstart berücksichtigt FENCE jedoch nicht und löscht alle vom Anwender definierten Worte.

FILL (a nl n2 --)
nl Bytes ab Adresse a werden mit dem Wert n2 belegt (niederwertiges Byte von n2).

FORGET
löscht Definitionen aus dem Dictionary. Anwendung in der Form:

FORGET Name

Alle Definitionen, die nach "Name" erfolgt sind, werden "vergessen". Dabei werden die jüngste Definition des angegebenen Namens sowie alle jüngeren Definitionen gelöscht und der Platz im Dictionary wieder freigegeben.

Falls jedoch der Eintrag des angegebenen Namens auf einer Adresse beginnt, die kleiner ist als der Wert von FENCE, erfolgt eine Fehlermeldung; es wird nichts gelöscht, und die Anfangsadresse des Namenseintrags wird auf den Stapel gelegt.

FORTH
ist der Name des Vokabulars, das von Anfang an zur Verfügung steht. Solange der Anwender kein anderes Vokabular definiert hat, werden alle seine Definitionen in das FORTH-Vokabular aufgenommen.

Hat der Anwender ein anderes Vokabular zum CONTEXT-Vokabular erklärt, und (nach DEFINITIONS) darin Funktionen eingetragen, so kann er durch den Aufruf FORTH wieder das FORTH-Vokabular zum CONTEXT-Vokabular erklären. (

Mit DEFINITIONS wird dann auch das CURRENT-Vokabular dem CONTEXT-Vokabular gleichgesetzt.

Der Anwender arbeitet dann wieder ausschließlich im FORTH-Vokabular; die Namen in den anderen Vokabularien sind zwar nicht gelöscht, werden aber weder angezeigt noch bei der Wörterbuchsuche gefunden. Erst wenn der Anwender den Namen des anderen Vokabulars aufruft, sind sie wieder verfügbar. *

HERE (-a)
legt den Wert a des Dictionary-Zeigers DP auf den Stapel, a ist die Adesse des ersten freien Bytes im Dictionary.

HEX (-)
schaltet für die Ein- und Ausgabe von Zahlen das hexadezimale Zahlensystem ein, indem BASE den Wert 16 erhält. Mit DEC kann auf dezimale Zahlendarstellung zurückgeschaltet werden, wie es beim Warm- oder Kaltstart von FORTH automatisch geschieht.

HLD (- a)
Während eine Zahl mit < # # #S für die Ausgabe aufbereitet wird, enthält die Anwendervariable HLD die Adresse, an der das zuletzt konvertierte Zeichen abgelegt wurde. Wird von < # initialisiert.

HOLD (n -)
kann während der Ausgabe-Aufbereitung von Zahlen zwischen < # und #> benutzt werden, um den bereits aufbereiteten Ziffern einen Dezimalpunkt oder ein anderes Zeichen anzufügen. Danach können die restlichen Ziffern aufbereitet werden, n ist der ASCII-Code des einzuschiebenden Zeichens.

I (- n)
Die Funktion I kann in DO-Schleifen benutzt werden, um den augenblicklichen Stand n des Schleifenzählers vom R-Stapel auf den C-Stapel zu kopieren.

ID. (a -)
gibt den Namen des FORTH-Wortes aus, dessen Namenseintrag bei Adresse a beginnt.

IF (n -)
Anwendung nur innerhalb von Wortdefinitionen in der Form:

Bedingung IF Wortfolge 1 ELSE Wortfolge2 ENDIF oder
Bedingung IF Wortfolge 1 ENDIF

Bei erfüllter Bedingung (allgemein: wenn IF zur Ausführungszeit einen von 0 verschiedenen Wert an oberster Stelle im Stapel vorfindet) wird Wortfolgel durchlaufen, andernfalls Wortfolge2, wenn vorhanden. In beiden Fällen wird die Verarbeitung nach ENDIF fortgesetzt. Schachtelung ist erlaubt.

Der Compiler übersetzt IF in einen bedingten Sprung OBRANCH auf die Stelle hinter ELSE oder (wenn ELSE fehlt) hinter ENDIF. Für ELSE wird ein unbedingter Sprung BRANCH auf die Stelle hinter ENDIF kompiliert.

IMMEDIATE (--)

setzt im jüngsten Namenseintrag des Wörterbuchs das "Immediate-Bit". Dies bewirkt, daß, wenn der so markierte Name später aufgerufen wird, seine Funktion sofort ausgeführt wird, auch wenn dies während des Kompilierens geschieht. IMMEDIATE kann also benutzt werden, um den Compiler durch weitere Funktionen zu erweitern.

IN (-- a)

Die Anwendervariable IN enthält als Wert die Nummer des Zeichens im Eingabepuffer, welches als nächstes zu verarbeiten ist. Die Nummer läuft von 0 bis maximal 79.

INS (- a)

Die als Konstante definierte Adresse INS gibt die Stelle im FORTH-System an, wo die zur Zeit angeschlossene Tracefunktion TRC aufgerufen wird. INS wird von der Funktion INSERT gebraucht.

Nicht in FIG-FORTH.

INSERT

Diese Funktion ermöglicht das Anschließen einer vom Anwender definierten Trace-Funktion (mit beliebigen Namen xxx) anstelle der eingebauten Trace-Funktion TRC. Nachdem der Anwender seine Trace-Funktion xxx definiert hat, kann er sie mit

INSERT xxx

an das FORTH-System anschließen. Sie wird dann, wenn FORTH-Funktionen bei eingeschaltetem Trace-Modus durchlaufen werden, vor jedem FORTH-Wort aufgerufen. Dabei wird die Adresse, an der das betreffende FORTH-Wort aufgerufen wird, an oberster Stelle des Stapels übergeben und muß in der Funktion xxx gelöscht werden.

Will der Anwender später wieder die originale Funktion TRC anschließen, so kann er dies mit

INSERT TRC

erreichen. Benötigt er dagegen die originale Funktion TRC nicht mehr, so kann er sie (nach Änderung von FENCE) löschen. Benötigt der Anwender überhaupt keine Trace-Funktion, so wird empfohlen, mittels

INSERT .

die Zahlenusgabefunktion anstelle von TRC anzuschließen. Damit wird erreicht, daß bei versehentlichem Drücken der -l-Taste das FORTH-System nicht abstürzt. Diesen "Pseudo-Tace" kann man dann mit t wieder ausschalten und das Programm mit ENTER fortsetzen.

Nicht in FIG-FORTH.

INTERPRET

Diese Funktion wird von QUIT aufgerufen, nachdem QUIT mittels QUERY eine neue Eingabezeichenfolge von der Tastatur geholt hat.

INTERPRET holt Wort für Wort mittels -FIND aus der Eingabe, bis diese abgearbeitet ist. Wird ein Wort im Wörterbuch gefunden, so wird es entweder kompiliert oder (kompiliert und) ausgeführt (je nach Immediate-Bit und Stellung des Schalters in STATE). Wird es nicht gefunden, so wird versucht, es als Zahl (gemäß der eingestellten Basis) zu verarbeiten. Ist auch das nicht möglich, so folgt Fehlermeldung ("— def"), und der Interpretiervorgang wird abgebrochen. Das Programm läuft auf irgendeine Weise (OK-Meldung, Fehlermeldung, Warmstart, ...) wieder auf QUIT, wo die nächste Eingabe-Zeichenfolge geholt und INTERPRET erneut aufgrufen wird.

KERN

Löscht (ohne Rücksicht auf FENCE) alle nicht unbedingt notwendigen FORTH-Funktionen bis auf den "Kern". Die gelöschten Funktionen werden auf der Kassette als FORTH-Quellcode in BASIC-Textzeilen zur Verfügung gestellt und können (nach Unterbrechung des laufenden FORTH-Kernsystems) mit dem BASIC-Kommando

CLOAD "PC-FORTH. SRC"

eingelesen werden (vgl. Anhang B).

Nach Kaltstart mittels DEF C können die Funktionen mit

40 <P

wieder an den FORTH-Kern angefügt werden. Dabei entsteht genau das originale PC-FORTH.

Statt dessen kann der Anwender aber auch (nach dem Einlesen des Quellcodes von der Kassette) den BASIC-Programmiermodus einschalten und Änderungen im FORTH-Quellcode vornehmen, bevor er die Funktionen (mittels <P im FORTH) wieder anfügt.

Wir haben so viele Funktionen als irgend möglich und sinnvoll aus dem "Kern" ausgelagert, um jedem Anwender bei der Gestaltung seines persönlichen FORTH-Systems freie Hand zu lassen. Wir müssen aber darauf hinweisen, daß Sie — falls Sie Programme mit anderen FORTH-Programmierern austauschen wollen — entsprechende Vorsicht und Zurückhaltung beim Ändern üben müssen.

Nicht im FIG-FORTH.

KEY (- n)
holt ein Zeichen von der Tastatur und legt seinen ASCII-Wert n auf den Stapel.

LATEST (- a)
Der Wert dieser Anwendervariablen ist die Anfangsadresse des jüngsten Namenseintrags im CURRENT-Vokabular.

LEAVE (-)
wird zwischen DO und LOOP oder zwischen DO und +LOOP angewendet und setzt die (durch DO im R-Stapel abgelegte) Schranke auf den (auch im R-Stapel abgelegten) augenblicklichen Wert des Schleifenzählers. Hierdurch kann man erreichen, daß die Schleife beim nächsten Aufruf von LOOP bzw. +LOOP verlassen wird.

LFA (a1 - a2)
rechnet eine Parameterfeldadresse in eine Linkfeldadresse um. Im PC-FORTH ist a2 = a1 - 4

LIT (-- n)

Findet der Compiler im Quellcode eine Zahl n, so legt er zuerst (die Codeadresse von) LIT und danach die (binärverschlüsselte) Zahl ab. Wird LIT später aufgerufen, so wird die hinter LIT gespeicherte 2Byte-Zahl n auf den Stapel gelegt.

LITERAL (n -)

übersetzt während des Kompilierens eine zuvor auf den Stapel gelegte Zahl so, daß später beim Durchlaufen des erzeugten Programmstücks diese Zahl auf den Stapel gelegt wird.
Typische Anwendung:

:...((Zahl n berechnen)) LITERAL ... ;

Das übersetzte Programmstück sieht im Prinzip wie folgt aus:

... LITn ...

(wobei natürlich anstelle von LIT die Codeadresse von LIT gespeichert ist und n binärverschlüsselt in den beiden Folgebytes steht).

So können z. B. Berechnungen, deren Ergebnisse schon zur Zeit des Kompilierens feststehen, auch während des Kompilierens ausgeführt werden. In der übersetzten Funktion steht (nach LIT) sofort das fertige, binärverschlüsselte Ergebnis, so wie es beim Aufruf der übersetzten Funktion auf den Stapel gelegt werden soll.

LOOP (-)

Anwendung innerhalb von Wortdefinitionen in der Form:

nl n2 DO Wortfolge LOOP

nl = Schranke; n2 = Anfangswert des Schleifenzählers

Die Wortfolge zwischen DO und LOOP wird sooft durchlaufen, bis der Schleifenzähler die Schranke nl erreicht oder überschritten hat, mindestens jedoch einmal.

Der Compiler speichert an der Stelle des DO-Aufrufs die zugehörige Laufzeitfunktion (DO), und an der Stelle von LOOP die Laufzeitfunktion (LOOP), gefolgt von der "relativen Rückwärtssprungadresse".

Zur Laufzeit transportiert (DO) die obersten beiden Zahlen n2 und nl aus dem C-Stapel auf den R-Stapel. Dann wird die Wortfolge zum ersten Mal durchlaufen.

LOOP addiert auf den im R-Stapel gespeicherten Schleifenzähler n2 eine 1 und vergleicht ihn dann mit der Schranke nl. Ist die Schranke noch nicht erreicht, so erfolgt Rücksprung entsprechend der hinter LOOP gespeicherten relativen Rückwärtssprungadresse.

Andernfalls löscht LOOP nl und n2 aus dem R-Stapel, und die Verarbeitung wird hinter der Stelle, wo die relative Rückwärtssprungadresse gespeichert ist, fortgesetzt.

Siehe auch +LOOP.

M* (nl n2 - d)
d = nl * n2

M/ (d nl - n2 n3)
n2 = Rest der Division d/nl
n3 = Quotient der Division d/nl

Das Vorzeichen von n2 stimmt immer mit dem Vorzeichen von d überein.

M/MOD (udl al -- a2 ud2)
führt gemischte Division aus:
udl = Dividend (vorzeichenfrei, doppelt lang)
al = Divisor (vorzeichenfrei, einfache Länge)
a2 = Rest (vorzeichenfrei, einfache Länge)
ud2 = Quotient (vorzeichenfrei, doppelt lang)

MAX (nl n2 - n3)
liefert das Maximum zweier Zahlen: n3 = Max(nl,n2)

MESSAGE (n --)
gibt die BASIC-Zeile mit der Nummer n + Wert von OFFSET als Meldung aus, wenn WARNING nicht den Wert 0 hat. Sollte die BASIC-Zeile nicht definiert sein, so wird die Meldung "~ Zeile" mit anschließender Zeilennummer ausgegeben.

Ist der Wert von WARING = 0, so wird n selbst ausgegeben (durch vorgestellte Anführungszeichen als Meldung gekennzeichnet).

MIN (nl n2 - n3)
liefert das Minimum zweier Zahlen: n3 = Min(nl,n2)

MINUS (nl - n2)
n2 = **-nl** (Zweierkomplement)

MOD (nl n2 -- n3)
n3 = Rest bei der Division nl/n2
(n3 hat immer das gleiche Vorzeichen wie nl)

NFA (al - a2)
errechnet aus einer Parameterfeldadresse die zugehörige Namensfeldadresse. (Benutzt TRAVERSE)

NOOP (-)
tut nichts (No Operation).

NUMBER (a - d)
konvertiert die ab a gespeicherte Zeichenfolge mit vorangehendem Längenbyte unter Berücksichtigung von BASE in eine doppeltgenaue Zahl d. Wie im Original FIG-FORTH festgesetzt, steht die höherwertige Zahlenhälfte an oberster Stelle im Stapel.

Negative Doppelzahlen werden im Zweierkomplement dargestellt. Wird während des Konvertierens ein Dezimalpunkt gefunden, so speichert die Funktion die Anzahl der Stellen nach dem Punkt in DPL; andernfalls erhält DPS den Wert-1.

Ist die Konversion nicht möglich, so erfolgt eine Fehlermeldung.

OFFSET (-a)
Der Wert dieser Anwendervariablen gibt die Nummer derjenigen BASIC-Zeile an, ab welcher die Fehlermeldungstexte gespeichert sind. Falls WARNING einen Wert < 1 hat, ist der Wert von OFFSET ohne Bedeutung.

In FIG-FORTH andere Bedeutung.

OIV (- a)
(Abkürzung für "Output interrupt variable")
Der Wert dieser Anwendervariablen enthält das Zeichen, mit dem der Anwender den letzten Ausgabestop oder EXPECT-Aufruf beendet hat.

OR (nl n2 - n3)
n3 = nl OR n2
Jedes Bit in n3 ist genau dann gesetzt, wenn in nl oder in n2 (oder in beiden Zahlen) das Bit gleicher Wertigkeit gesetzt ist.

OUT (- a)

Diese Anwendervariable enthält einen Wert, der bei jedem (mit EMIT) ausgegebenen Zeichen um 1 erhöht wird.

In PC-FORTH zählt OUT modulo 24, d. h. nach 24 wird der Ausgabepuffer von EMIT ausgegeben und OUT auf Null gesetzt.

OVER (nl n2 -- nl n2 nl)

legt eine Kopie der zweitobersten Zahl auf den Stapel.

P >

beendet die Eingabe aus dem BASIC-Programmspeicher (vgl. <P).

Nicht in FIG-FORTH.

PÄD (- a)

Diese Funktion liefrt die Anfangsadresse eines Speicherbereichs, der als kurzzeitiger Zwischenspeicher benutzt werden kann. Die von PÄD gelieferte Adresse liegt stets 68 Bytes nach HERE, ändert sich also, wenn Worte definiert oder gelöscht werden.

PFA (al - a2)

rechnet die Namensfeldadresse eines Namenseintrags in die Parameterfeldadresse um. (Benutzt TRAVERSE)

PICK (nl -- n2)

kopiert die nl-te Zahl vom Stapel an die oberste Stapelposition.
n2 ist die nl-te Zahl im Stapel, wobei von oben ab 1 gezählt wird, nl jedoch nicht mitgerechnet.

PICK püft nicht, ob der gesuchte Eintrag noch innerhalb des Stapelbereichs liegt.

QUERY

fordert eine Zeichenfolge von der Tastatur an. PC-FORTH wechselt zu diesem Zweck ins BASIC und fordert die Eingabe mittels INPUT I$(0) an. Daher können Tippfehler während der Eingabe, wie von BASIC gewohnt, korrigiert werden. Die Eingabe wird mit ENTER abgeschlossen. Die Startadresse des eingegebenen Textes liefert die Anwendervariable TIB.

I$(0) ist als 80Byte-Textfeld dimensioniert. Seine Anfangsadresse steht in der Anwendervariablen TIB; dorthin wird sie bei Kaltstart aus ORIGIN + 20 gebracht.

Der Eingabepuffer TIB darf in Anwenderprogrammen normalerweise nicht verändert werden, daher ist QUERY in Anwenderprogrammen nur mit großer Vorsicht zu gebrauchen.

QUIT

löscht den R-Stapel und beendet ggf. die Eingabe aus BASIC-Programmzeilen. Ein etwa bestehender Kompilierzustand wird jedoch nicht beendet. QUIT gibt keine Meldung aus.

R (- n)

kopiert die oberste Zahl aus dem R-Stapel auf den C-Stapel. (Somit wirkt R exakt wie I.)

R > (- n)

tansportiert die oberste Zahl aus dem R-Stapel auf den C-Stapel. (Der oberste R-Stapel-Eintrag wird also gelöscht.)

RO (- a)

Diese Anwendervariable enthält den Anfangswert des R-Stapelpegels (Adresse, bei der der R-Stack beginnt). Vgl. Anhang B.

REPEAT (--)

Wird in der Form

BEGIN .. WHILE .. REPEAT

benutzt. Siehe WHILE.

ROT (nl n2 n3 - n2 n3 nl)

transportiert die drittoberste Zahl im Stapel auf die oberste Stelle, wobei die oberste und zweitoberste Zahl nach unten geschoben werden.

RP (- a)

Diese Anwendervariable enthält den Pegel des R-Stapels.

RP!

schreibt den Wert der Anwendervariablen RO in die Anwendervariable RP und initialisiert damit den R-Stapel.

RPICK (nl - n2)

kopiert die nl-te Zahl des R-Stapels (von oben ab 1 gezählt) auf den C-Stapel, wobei nl im C-Stapel überschreiben wird. (n2 wird im allgemeinen die Bedeutung einer Adresse haben.)

S->D (n -- d)

wandelt eine einfache Zahl in eine ihr gleiche doppeltlange Zahl um. (Je nach Vorzeichen von n ist die obere Hälfte der 4Byte-Doppelzahl entweder Hex 0000 oder Hex FFFF.)

SO (- a)

Diese Anwendervariable enthält den Anfangswert des C-Stapelpegels (Adresse, an der der Stapel beginnt). Vgl. Anhang B.

SIGN (n d - d)

ist während der Ausgabekonversion einer Zahl anwendbar, also zwischen <# und #>. Setzt ein Minuszeichen vor die bisher konvertierte Zeichenfolge, falls n<0 ist. Dabei wird n gelöscht, aber die 4Byte-Zahl d bleibt unverändert, weil sie während der Konversion noch gebraucht wird.

SMUDGE

wechselt im jüngsten Namenseintrag das "smudge bit" (im Längenbyte) von 0 in 1 bzw. von 1 nach 0. Solange eine Definition nicht abgeschlosen ist, darf ihr Name von -FIND normalerweise nicht gefunden werden. Dies wird durch das gesetzte "smudge-bit" erreicht. Bei rekursiven Programmen ist es jedoch nötig, daß eine Funktion sich selbst aufruft. Dazu wird in der Definition vor dem rekursiven Aufruf SMUDGE eingesetzt.

SP! (löscht Stapel)

schreibt den Wert der Anwendervariablen SO in das Register S des Mikroprozessors und setzt damit den Stapelpegel auf seinen Anfangswert.

SP© (- a)
legt den augenblicklichen Stapelpegel als Adresse a auf den Stapel, (a ist also die Adresse, die dem Stapelpegel entsprach, bevor a gestapelt wurde.)

SPACE (-)
gibt ein Leerzeichen aus.

SPACES (n --)
gibt n Leerzeichen aus.

STAT®
wirkt genau wie STATE © (wurde nur zwecks Optimierung eingeführt).

Nicht in FIG-FORTH.

STATE (- a)
Der Wert dieser Anwendervariablen ist 0 während des Ausführungszustands und Hex CO während des Kompilierens.

SWAP (nl n2 -- n2 nl)
vertauscht die beiden obersten Stapeleinträge.

TASK
wird gerne als Grenze zwischen verschiedenen Gruppen zusammengehöriger Anwenderfunktionen definiert. Mit FORGET TASK kann der Anwender dann jeweils die jüngste(n) Gruppe(n) löschen. Dient nur der Übersichtlichkeit, wird nicht aufgerufen.

THEN
wirkt genau wie ENDIF. (Nicht verwechseln mit dem THEN aus BASIC!)

TIB (- a)
(Abkürzung für "terminal input buffer")
Der Wert dieser Anwendervariablen ist die Anfangsadresse des Eingabepuffers. Er muß mit der BASIC-Variablen I$(0) identisch sein.

*

TIC (-- a)

Anzuwenden in der Form:

TIC name

Liefert die Parameterfeldadresse (PFA) der Definition des Wortes "name". Ab a ist also im allgemeinen der kompilierte Code der Wortdefinition gespeichert.

Heißt in FIG-FORTH '

TOB (-a)

(Abkürzung für "terminal Output buffer")

Der Wert dieser Anwendervariablen ist die Anfangsadresse des Ausgabepuffers. Er muß mit der BASIC-Variablen O$(0) identisch sein.

Nicht in FIG-FORTH.

TOGGLE (a n -)

Von n wird nur das niederwertige Byte vewendet. Es wird mit dem auf Adresse a stehenden Byte bitweise mit "XOR" verknüpft. Das Resultatbyte wird in a gespeichert.

TRACE

schaltet den Traceschalter ein. Dies bewirkt, daß vor jedem aufzurufenden FORTH-Wort die Funktion TRC aufgerufen wird. Somit wirkt TRACE wie 1 (doch kann 1 nur während eines Ausgabesteps gegeben werden).

Mit TROFF oder T kann der Traceschalter wieder ausgeschaltet und der Programmlauf fortgesetzt werden.

TRACE und TROFF sollten nur in Wortdefinitionen gegeben werden. Sie funktionieren zwar auch bei manueller Eingabe, doch werden dann der Compiler und Interpreter selbst schrittweise überwacht.

EXECUTE kann mit TRC nicht richtig überwacht werden; darum ist vor EXECUTE unbedingt TROFF oder T zu geben.

Nicht in FIG-FORTH.

TRAVERSE (al n - a2)

wird in NFA und PFA benutzt, um entweder das nächste vorhergehende oder das nächstfolgende Byte mit gesetztem Bit 7 zu finden, n gibt die Richtung des Suchens an. n ist entweder + 1 oder -1 und wird auf al sooft addiert, bis der Inhalt der sich ergebenden Adresse wieder ein Byte mit Bit 7 = 1 ist. Die so entstandene Adresse wird als a2 hinterlassen.

TRC (a -)

Bei eingeschaltetem Trace wird vor jedem FORTH-Wort-Aufruf ein Aufruf der Funktion TRC eingeschoben. Die Adresse a, ab welcher der betreffende Funktionsaufruf (d. h. die Codeadresse des aufzurufenden FORTH-Wortes) gespeichert ist, wird der Funktion TRC als oberster Stapeleintrag übergeben.

TRC gibt nun folgendes aus:

die Adresse a
den Namen des aufzurufenden FORTH-Worts
den obersten Stapeleintrag
den zweitobersten Stapeleintrag und
ein CR

Die Zahlen werden in vorzeichenloser Hex-Darstellung ausgegeben.

ENTER oder I setzt den Trace fort; T schaltet den Trace ab, so daß der Programmlauf wieder normal fortgesetzt wird.

TRC wurde ohne großen Aufwand programmiert; die vorhandenen Schwächen dürften sich aber normalerweise in Kauf nehmen lassen. (Falls nicht, vgl. INSERT)

TRC hat folgende Nachteile:

— Die Überwachung von EXEC wird nicht richtig ausgeführt. Sobald EXEC ausgegeben wird, muß der Trace ausgeschaltet werden.
— Es ist nicht möglich, Variablen oder sonstige Speicherinhalte mit Hilfe zusätzlich eingeschobener FORTH-Funktionen anzuschauen. (Zu diesem Zwecke wäre es nötig gewesen, z. B. den Inhalt des Eingabepuffers sicherzustellen.)
— Der Inhalt des Ausgabepuffers wird von der Trace-Ausgabe überschrieben.
— TRC überwacht auch die Funktionen des FORTH-Systems.
Deshalb ist es im allgemeinen ratsam, vor Worten wie . oder D. den Trace abzuschalten.

Nicht in FIG-FORTH.

TROFF

schaltet den Trace ab. Es hat somit die gleiche Wirkung wie t.

Nicht in FIG-FORTH.

TYPE (a n -)

gibt den ab a gespeicherten Text der Länge n aus.

U* (al a2 - ud)

ud = al * a2 (U * ist die Multiplikationsroutine, auf die alle anderen Multiplikationsbefehle zurückgreifen, und wird mit Abstand am schnellsten ausgeführt.)

U. (a -)

gibt den obersten Stapeleintrag vorzeichenfrei unter Berücksichtigung von BASE aus.

U/ (ud al -- a2 a3)

dividiert ud durch al, a3 ist der Quotient, a2 der Divisionsrest.
(U/ ist die Divisionsroutine, auf die alle anderen Divisionsbefehle zurückgreifen, und wird daher mit Abstand am schnellsten ausgeführt.)

U< (al a2 -- n)

n = 1, wenn für die beiden vorzeichenfreien Zahlen al und a2 die Beziehung al < a2 gilt, sonst n = 0.

UNTIL (n -)

wird innerhalb von Definitionen in folgender Form angewandt:

BEGIN Wortfolge Bedingung UNTIL

Die Wortfolge wird solange wiederholt, bis die Bedingung erfüllt ist (d. h. bis UNTIL als obersten Stapeleintrag eine von 0 verschiedene Zahl vorfindet).

Der Compiler übersetzt UNTIL als (Codeadresse von) OBRANCH mit darauffolgender relativer Rückwärtssprungadresse.

USER (n -)

definiert eine Anwendervariable. Aufruf in folgender Form:

n USER Name

wobei n den Abstand zur Anfangsadresse des Anwenderbereichs (in Bytes) angibt.

Anwendervariable und Anwenderbereiche haben ihre Bedeutung nur in Multitaskingsystemen, in denen mehrere Anwender gleichzeitig mit unterschiedlichen Anwenderbereichen arbeiten.

Da der Zugriff auf gewöhnliche Variable wesentlich schneller ist als der Zugriff auf Anwendervariable, wurde der Anwenderbereich im PC-FORTH so gelegt, daß höchstens 7 neue Anwendervariable definiert werden können. Sollte diese Einschränkung unerwünscht sein, so ändere man vor dem Kaltstart die auf ORIGIN + 14 stehende Anfangsadresse des Anwenderbereichs so, daß Speicherplatz im gewünschten Umfang zur Verfügung steht (vgl. Anhang B).

Maximal 128 Anwendervariable können insgesamt (im ganzen FORTH-System) definiert sein. (Ebenso wie in FiG-FORTH.)

VARIABLE (n -)
wird in der Form

n VARIABLE Name

benutzt, um eine Variable mit dem angegebenen Namen zu definieren und ihr die oberste Zahl n im Stapel als Anfangswert zuzuweisen, n wird im Stapel gelöscht. Späterer Aufruf des Namens bewirkt, daß die Adresse, ab der der jeweilige Wert der Variablen gespeichert ist, auf den Stapel gelegt wird. Der Wert kann dann mit ©auf den Stapel geladen oder mit! verändert werden.

VOC-LINK (- a)
Der Wert dieser Anwendervariablen zeigt auf den Namenseintrag des jüngsten (d. h. zuletzt definierten) Vokabulars. Mit seiner Hilfe verkettet das FORTH-System alle Vokabularien.

VOCABULARY
Anzuwenden in der Form:

VOCABULARY Name IMMEDIATE

definiert ein Vokabular mit dem angegebenen Namen. Späterer Aufruf des Namens macht das Vokabular zum "Context-Vokabular", d. h. zu demjenigen Vokabular, in dem INTERPRET alle eingegebenen Worte zuerst sucht.

In PC-FORTH werden (wie in FIG-FORTH) die Vokabularien so verkettet, daß nach erfolglosem Durchsuchen eines Vokabulars das Suchen in demjenigen Vokabular fortgesetzt wird, in dem das erfolglos durchsuchte Vokabular definiert wurde. Dies hat zur Folge, daß die Worte des FORTH-Vokabulars immer erreichbar sind.

VLIST (-)

gibt alle Namen aus, die im Context-Vokabular definiert sind. Nach Abarbeitung eines Vokabulars folgt jeweils dasjenige Vokabular, indem das abgearbeitete Vokabular definiert wurde.

Die Ausgabe kann nach einem Ausgabestop mit der Taste • abgebrochen werden.

W, (n -)

arbeitet wie WORD, legt jedoch zusätzlich das vom Zeichen mit dem ASCII-Code n begrenzte Wort als Zeichenfolge ab HERE im Dictionary ab, wobei DP (HERE) entsprechend erhöht wird. Nicht in FIG-FORTH.

WARNING (- a)

Ist der Wert dieser Anwendervariablen 1, werden Fehlermeldungen als Texte ausgegeben.
Ist er 0, werden nur Fehlernummern ausgegeben.
Ist er - 1 , erfolgt im Fehlerfall stets ABORT.

Die Fehlermeldungstexte stehen in BASIC-Zeilen und können (im BASIC-Programmiermodus) vom Anwender nach Wunsch geändert werden.

Die BASIC-Zeilennummern der Fehlermeldungstexte errechnen sich aus Fehlernummer + Wert von OFFSET; werden die Texte in einen anderen Zeilennummernbereich verlegt, Ist dei Wert von OFFSET entsprechend zu ändern.

Zwecks Platzersparnis können alle Fehlertexte gelöscht werden. Dazu ist der Wert von WARNING auf 0 (oder -1) zu setzen.

WHILE (n -)

Anwendung in der Form:

BEGIN Wortfolgel Bedingung WHILE Wortfolge2 REPEAT

Wortfolgel wird mindestens einmal durchlaufen. Ansonsten werden Wortfolgel und Wortfolge2 solange wiederholt, wie die Bedingung erfüllt ist (d. h. wie WHILE an oberster Stelle des Stapels einen von 0 verschiedenen Wert vorfindet).

Danach wird die Verarbeitung hinter REPEAT fortgesetzt.

Der Compiler legt für BEGIN keinen Code ab, sondern merkt sich nur die Stelle. Für WHILE speichert er einen bedingten Sprung auf die nach REPEAT eingegebene Wortfolge (also OBRANCH und relative Vorwärtssprungadresse), und für REPEAT einen unbedingten Rücksprung auf Wortfolgel (also BRANCH und relative Rückwärtssprungadresse).

WIDTH (~ a)

Der Wert dieser Anwendervariablen gibt an, mit welcher Maximallänge Namen ins Wörterbuch eingetragen werden. Standardmäßig enthält WIDTH den Wert 31. Ein niedrigerer Wert von WIDTH spart Speicherplatz, allerdings werden Forthnamen dann nur noch mit geringerer Zeichenzahl unterschrieben. Manche FORTH-Systeme arbeiten mit WIDTH = 3.

WORD (n -)

liest ein Wort, das vom Zeichen mit dem ASCII-Code n begrenzt wird, aus der Eingabezeichenfolge in den Speicher ab Adresse HERE.

Dabei wird zunächst das Längenbyte, dann die Zeichenfolge gefolgt von ein oder mehreren Leerzeichen abgelegt.

WLIST (a -)

dient der Auflistung des vom Compiler erzeugten FORTH-Codes. Typische Anwendung in der Form:

TIC Name WLIST

gibt ab Adresse a (TIC Name) fortlaufend die Adressen, die dort gespeicherten Codeadressen und die zu diesen gehörigen FORTH-Namen aus.

Beim Ausgabestop kann WLIST mit • abgebrochen werden.

WLIST arbeitet analog zu einem Disassembler und steht daher vor ähnlichen Problemen bei der Rückübersetzung des Codes: Im Code gespeicherte Daten werden unter Umständen fehlinterpretiert und können bei ungerader Anzahl von Datenbytes zum "Aushaken" von WLIST und falschen Ausgaben führen.

XOR (nl n2 - n3)
n3 = nl **XOR** n2. Zwischen nl und n2 wird bitweise ein Exklusiv-Oder (entweder-oder) durchgeführt.

[**(FIG-FORTH)**
heißt in PC-FORTH ((

[COMPILE] **(FIG-FORTH)**
heißt in PC-FORTH ((COMPILE))

I **(FIG-FORTH)**
heißt in PC-FORTH))

Kapitel 4

Beispiel: Nützliche Definitionen

Im folgenden finden Sie einige nützliche Wortdefinitionen, die Ihnen die Arbeit mit dem Forth-System erleichtern. Diese Definitionen sollen allerdings in erster Linie der Anregung dienen, wie Sie PC-FORTH weiter ausbauen können.

4.1 Druckersteuerung

Zunächst einige Definitionen, die den Drucker betreffen:

```
200": LON(-)
205"  (schaltet Drucker ein)
210"  1 !: "L";   (L=l)
215"
220": LOFF(-)
225"  (schaltet Drucker aus)
230"  0!:"L";    (L = 0)
235"
240"HEX
245": CSIZE (n--)
250"  79F4 C! ;
255"DEC
260": COLOR ( n - )
265"  !:"C"       (C = n)
270"  26 BAL ;    ( GOTO 26)
275"P>
```

Zeile 26 im BASIC wird zur Verwendung durch COLOR wie folgt eingegeben:

```
26 COLOR C:GOTO"H"
```

Mit LON und LOFF können Sie den Drucker programmgesteuert ein- und ausschalten. Die eigentliche Druckersteuerung erfolgt über die BASIC-Variable L in den Zeile 20 bis 25 des BASIC-Teils vom Forth-System. CSIZE und COLOR entsprechen den BASIC-Kommandos.

4.2 Verschiedenes

Die beiden folgenden Befehle dienen der Speicherverwaltung. MEM liefert die augenblicklich belegten Speicherbereiche. DELETE dient zum Löschen von BASIC-Zeilen im Programmspeicher.

```
300"HEX
305": MEM ( - )
310"  (zeigt Speicherbelegung)
315"  .'FORTH: -    0 +ORIGIN 6 .R HERE 6 .R CR
320"  ." STACK: "    SP© 6 .R SO 6 .R CR
325"  ." BASIC: "     7865 © 6 .R 7867 © 6 . R C R ;
330"
335": DELETE ( n - )
340"  (löscht alle BASIC-Zeilen ab Nummer n)
345"  B# 3 - DUP   ( 1. Byte der Zeilennummer)
350"  FFSWAPC!    ( Endmarker FF)
355"  7867 ! ;    ( zeigt auf Programmspeicherende)
360" P >
```

Die folgenden Worte dienen der Zusammenarbeit zwischen BASIC und PC-FORTH. Mit GOTO kann eine BASIC-Zeile mit beliebiger Nummer angesprungen werden, mit dem Befehl BASIC kann ein BASIC-Programm, das direkt an den Forth-Quelltext anschließt, gestartet werden.

```
400": GOTO ( n - )
405"  (springt in die BASIC-Zeile n, Rückkehr mit GOTO "H")
415"  !:"G"40BAL;     ( n nach G und Sprung zur Zeile 40)
```

Zeile 40 wird so eingegeben:

```
40 GOTO G
```

```
420": BASIC ( - )
425"  ( wechselt von Forth ins BASIC)
430"  DROP 0 IN !   ( Relikte von <P beseitigen)
435"  BLK ©         ( augenblickliche Zeilennummer )
440"  0 BLK !       (stornieren von <P )
445"  GOTO ;
```

Zeile 41 wird folgendermaßen eingegeben:

```
41"FORTH":I$(0) = STR$(I) + " <P":GOTO "H"
```

Jetzt können BASIC- und Forth-Anweisungen folgendermaßen gemischt werden:

```
1000 PRINT'Wir sind im BASIC!"
1010 1= 1020:GOTO"FORTH"
1020 " ." Wir sind im FORTH!" CR
1030 " ( Weitere Forth-Befehle)
1040 "BASIC "BEEP 1
1050 PRINT'Wir sind wieder im BASIC!"
```

Wird der Variablen I eine Zeilennummer zugewiesen und GOTO"FORTH" durchgeführt, so wird der Forth-Quellcode ab der angegebenen Zeilennummer ausgeführt. Mit dem Forth-Wort BASIC wird dann noch in derselben Zeile auf BASIC-Betrieb zurückgeschaltet.

4.3 Stringvariable

Mit den folgenden Definitionen werden Stringvariable in Forth realisiert.

```
400": 1- ( n - n - 1 )
405"  1 - ;  (Abkürzung)
410"
415": SRCH(a-an)
420"  ( bestimmt die Länge n eine Zeichenfolge,)
425"  ( die ab Adresse a gespeichert ist und)
430"  ( mit 0 endet)
435"  DUP
440"  BEGIN
445"     DUP C© SWAP 1 + SWAP
450"  0=  UNTIL
455"  SWAP - 1- ;
460"
470": STRING(n-)
475"  ( Definiert und dimensioniert Stringvariable)
480"  ( der Länge n. Beim Aufruf der Stringvariablen )
```

```
485"  ( kommt Adresse a und Länge n auf den Stapel)
490"  <BUILDS
495"     ABS 255 MIN 1 MAX     ( Maximallänge 255, Minimal 1)
500"     DUP  C,               ( Längenbyte eintragen)
505"     0 DO 32 C, LOOP       ( Mit SPACES auffüllen )
510"     0 C,                  ( 0 als Endzeichen anfügen )
515"  DOES>
520"  1+ DUP SRCH ;            ( Anfangsadresse und Länge stapeln)
525"
530"0 VARIABLE IB 80 ALLOT     ( Pufferspeicher von 80 Zeichen für)
535"                           ( die Stringeingabe)
540": (^) ( - - a n)
545"  ( Laufzeitfunktion zu ^ legt Anfangsadresse und Länge des)
550"  ( im Programm gespeicherten Strings auf den Stapel)
555"  R COUNT                  ( Anfangsadresse und Länge stapeln)
560"  DUP 1+ R> + >R ;         ( gespeicherten String überspringen)
565"
570": ^ ( -- a n)
575"  ( wird in der Form ^ Zeichenfolge^ angewendet, um Strings)
580"  ( einzugeben. )
585"  94                       ( ASCII-Wert von ^ )
590"  STATE @                  ( In Definition? )
595"  IF
600"   COMPILE (^)             ( Laufzeitfunktion kompilieren)
605"   WORD                    ( bis ^ einlesen und)
610"   HERE C@ 1+ ALLOT        ( im Programmcode abspeichern)
615"  ELSE                     ( Keine Definition, direkt ausführen)
620"    WORD                   ( bis ^ einlesen)
625"    HERE COUNT             ( Anfangsadresse und Länge stapeln)
630"    IB SWAP ROT OVER       ( nach Puffer IB kopieren)
635"    IB SWAP 1+ CMOVE
640"    2DUP + 0 SWAP C!       ( 0 als Endzeichen anfügen)
645"  ENDIF ;
650"IMMEDIATE
655"
660": S! ( a1 n1 a2 n2 --)
665"  ( Speichert String bei a1 der Länge n1 in Stringvariable)
670"  ( bei a2)
```

```
675" ( Wird in der Form" String" Stringvariable S! verwendet)
680" DROP DUP 1- C@          ( Maximallänge aus Variable)
685" ROT MIN 1 MAX           ( nicht länger als Maximallänge)
690" 2DUP + 0 SWAP C!        ( Endzeichen 0 eintragen)
695" CMOVE ;                 ( davor die Zeichenfolge speichern)
```


```
675" ( Wird in der Form" String" Stringvariable S! verwendet)
680" DROP DUP 1- C©          ( Maximallänge aus Variable)
685" ROT MIN 1 MAX           ( nicht länger als Maximallänge)
690" 2DUP + 0 SWAP C!        ( Endzeichen 0 eintragen)
695" CMOVE ;                 ( davor die Zeichenfolge speichern)
```


Diese Definitionen gehen auf einen Vorschlag von R. Deane zurück, der 1980 in Dr. Dobbs "Journal of Computers" erschien, und gehören mittlerweile fast zum Standard.

(Stringvariablen werden in der Form:

    n STRING Variablenname

definiert, wobei n die Maximallänge der darin speicherbaren Zeichenfolge angibt. Bei der Definition werden Stringvariablen mit Leerzeichen aufgefüllt.
Beim Aufruf einer Stringvariablen, wird die Anfangsadresse und die Länge der augenblicklich gespeicherten Zeichenfolge auf den Stapel gelegt. Daher kann die gespeicherte Zeichenfolge z.B. direkt nach dem Variablenaufruf mittels TYPE ausgegeben werden.

Strings werden anstelle von Anführungszeichen mit" eingegeben. Sie können mit S! in einer Stringvariable gespeichert werden:

    * Text" Stringvariable S!

Stringoperationen, die mit den gestapelten Adressen und Längen operieren, können jetzt ohne größeren Aufwand definiert werden.

Beispiel:

```
20 STRING A$
" TEXTBEISPIELE" A$ S!
A$ TYPE CR
```

Anhang A

Fehlermeldungen

Die Texte der Fehlermeldungen stehen in BASIC-Zeilen und können von Ihnen jederzeit geändert werden.

Die Fehler werden im wesentlichen nur vom äußeren Interpreter überprüft. Tritt beim Aufrufeines Wortes während der Ausführung ein Fehler auf, so wird er im allgemeinen erst nach Beendigung des Aufrufs (wenn überhaupt) registriert.

"~def	Wort nicht definiert, auch nicht als Zahl interpretierbar Beispiel: 1234XYZ, solange Sie dieses Wort nicht definiert haben.
"leer	Der Stapel ist leer Beispiel: DEF F (Warmstart) und dann DROP
"Zeile ~def	BASIC-Zeilennummer nicht vorhanden.
"~ neu	Name bereits definiert, die neue Definition wurde ausgeführt und gilt jetzt anstelle der alten. Beispiel: : DUP 1 PICK ;
"voll	Im Stapel oder Wörterbuch ist nicht mehr genügend Platz vorhanden.
"nur :Def	Das Wort darf nur innerhalb von Wortdefinitionen verwendet werden. Beispiel: DO
"~compilierbar	Das Wort darf nicht in Wortdefinitionen verwendet werden. Beispiel: VARIABLE
"passt nicht	Unerlaubte Überkreuzung von Kontrollstrukturen Beispiel: IF...WHILE...LOOP
"strukturfehler	; trotz nicht abgeschlossener Kontrollstruktur Beispiel: IF ;

"< fence — Es wurde versucht mit FORGET ein Namen zu löschen, der durch FENCE geschützt ist.
Beispiel: FORGET :

"~ ladend — das Wort darf nur aus BASIC-Zeilen eingegeben werden, nicht direkt von der Tastatur.
Beispiel: P>

"basic — BASIC hat einen Fehler an Forth gemeldet.
Beispiel: !: "A/3"

Beim Einlesen aus BASIC-Zeilen mit <P, wird bei jedem Fehler die Zeilennummer mit vorausgehenden # angezeigt. Wird eine negative Zeilennummer angezeigt, so wurde vergessen, das Einlesen mit P> zu beenden.

Anhang B

Speicherbereiche und Kaltstartwerte

Wird PC-FORTH installiert, so wird mit einem entsprechendem NEW-Befehl der Beginn der des BASIC-Programmspeichers heraufgesetzt und PC-FORTH an den Speicherbeginn (+ &C6) geladen.

Es ergibt sich folgende Speicheraufteilung:

RAM-Beginn		Reservebelegung der Tastatur	
RAM-Beginn + 198	ORIGIN DP (HERE) CSP (SP@) S0	PC-FORTH Neue Definitionen Stapel (C-Stack)	↓ ↑
	R0	Return-Stack	↑
	UV	Anwendervariablen (User Variables)	
RAM-ENDE	TOB TIB	BASIC-Programmspeicher BASIC-Variablen O$(0) I$(0)	↓ ↑

Die genaue Speicherbelegung kann mit dem MEM-Befehl aus Kapitel 4 bestimmt werden.

Die vom Forth-System verwendeten Speicherbereiche werden bei jedem Kaltstart von PC-FORTH neu festgelegt, indem die Adresswerte aus bestimmten Speicheradressen ab ORIGIN entnommen werden. Sie können in diese Speicheradressen andere Initialwerte vorgeben und somit die Speicheraufteilung des Forth-Systems ändern.

Im einzelnen sind folgende Adressen beim Kaltstart von Bedeutung:

ORIGIN	+ 0	Sprungbefehl zur Kaltstartroutine	
	+ 4	Sprungbefehl zur Warmstartroutine	
	+ 6	Sprungbefehl zur Heißstartroutine	
	+ 8	Implementationsattribut &01 Ol 00 OC	
	+ 12	Oberstes Wort im FORTH-Vokabular	
	+ 14	UV Beginn des Anwendervariablenbereichs	
	+ 16	Beginn des C-Stapels	SO
	+ 18	Beginn des R-Stapels	R0
	+ 20	Eingabepuffer (Adresse von I$(0)	TIB
	+ 22	Ausgabepuffer (Adresse von O$(0)	TOB
	+ 24	Maximale Namenslänge (= 31)	WIDTH
	+ 26	Schalter für Fehlermeldungen (= 0)	WARNING
	+ 28	Zeilennummer der ersten Meldung	OFFSET
	+ 30	Oberstes gegen FORGET geschütztes Wort	FENCE
	+ 32	Oberstes Wort im Wörterbuch	DP (HERE)
	+ 34	Vokabularverkettung	VOC-LINK
	+ 36	CPU-Name als Doppelzahl	

Links neben den Werten stehen die Anwendervariablen, die mit den bei diesen Adressen eingetragenen Werten beim Kaltstart initialisiert werden.

Durch Änderung des entsprechenden Kaltstartwerts können z.B. die beiden Stapel in einen anderen Adressbereich verlegt werden.

Durch entsprechende Änderung von ORIGIN + 12, ORIGIN + 30 und ORIGIN + 32 kann das Kaltstart FORTH-System um neue Anwenderdefinitionen erweitert werden. Diese brauchen dann nach Kaltstart nicht mehr zu kompiliert werden.

Wird dabei das FORTH-System von ORIGIN bis HERE mit CSAVEM auf Kassette gespeichert, so stehen die neu aufgenommenen Worte sofort nach dem Laden wieder zur Verfügung.

Der BASIC-Programmspeicherbeginn kann ohne weitere Änderungen am FORTH-System nach oben verlegt werden. Lediglich die Variablen I$(0) und O$(0) müssen stets bei den Adressen TIB und TOB liegen. Daher sollten sie immer zuerst nach CLEAR oder NEW dimensioniert werden.

Mit dem Befehl KERN kann das FORTH-Vokabular um viele Befehle gekürzt werden. Auf der mitgelieferten Kassette sind diese löschbaren Befehle unter dem Namen PC-FORTH.SRC als Quellcode mitgeliefert und können somit auch vom Anwender geändert werden.

Da dafür im BASIC-Speicher mehr als 3K benötigt werden, muß gegebenenfalls die Speicheraufteilung mit NEW und Vorgabe neuer Kaltstartwerte für UV, SO, RO geändert werden, falls weniger als 3K für das BASIC zur Verfügung stehen.

Anhang C

Wörterbuch, Compiler, Interpreter

C.l Quellcode und Pseudocode

Der Compiler hat die Aufgabe, den Forth-"Quellcode", wie ihn der Benutzer eingibt, in einen internen Zwischencode ("Pseudocode") zu übersetzen und diesen im Speicher abzulegen.

Der Quellcode besteht aus einer Folge von Forthworten in der Textform, die durch Leerzeichen voneinander getrennt sind, und vom Anwender über die Tastatur eingegeben werden.

Der interne Zwischencode (Pseudocode) besteht aus einer Folge von Zahlenwerten, die vom Compiler im Speicher abgelegt werden. Anstelle eines jeden Forthwortes legt der Compiler nämlich einen Adresswert als Code ab. Jedes Codewort beansprucht also 2 Bytes (16 Bit).

Der abgelegte Adresswert zeigt auf das sogenannte Codefeld der zum eingebenen Namen gehörigen Definition. Dieser Adresswert wird deshalb auch Codefeldadresse (CFA) genannt.

C.2 Innerer und äußerer Interpreter

Der sogenannte äußere Interpreter ist dafür verantwortlich, die von der Tastatur eingegebenen Forthworte zu interpretieren und auszuführen. Zum Beispiel schaltet er den Compiler ein, sobald ein Doppelpunkt eingegeben wird.

Er sucht das Wörterbuch nach den eingegebenen Namen ab. Sobald er einen Namen gefunden hat, ruft er das nach dem Namen gespeicherte compilierte Programm auf.

Der innere Interpreter ist dafür verantwortlich, Programme, die als Pseudocode vorliegen, auszuführen. Er holt sich von der augenblicklichen Programmadresse den dort abgelegten Pseudocode und ruft das zur Ausführung nötige Maschinenprogramm auf.

Der äußere Interpreter und der Compiler arbeiten beide relativ langsam, da sie für jeden Befehl erst das ganze Wörterbuch absuchen müssen.

Der innere Interpreter arbeitet dagegen sehr schnell, da er gleich die richtigen Adressen bekommt und nicht erst suchen muß.

(Ein Unterprogrammaufruf in der Maschinensprache dauert 30 Taktzyklen bzw. 23 us, der innere Interpreter braucht für den Aufruf des zum Befehl gehörigen Maschinenprogramms 67 Taktzyklen bzw. 52 us. Daher arbeiten die in den Pseudocode kompilierte Forthprogramme oft fast so schnell, wie manche Compiler, die in Maschinencode überstetzen und starken Gebrauch von Unterprogrammen machen. Umgekehrt blähen die schnelleren Compiler die Programmlänge oft enorm auf, sodaß der Forth-Code einen guten Kompromiß darstellt.)

C.3 Wörterbuch und Compiler

Der Namenseintrag eines Forthwortes ist folgendermaßen aufgebaut:

LG	Name	LK	CD	Parameter
t		T	T	t
NFA		LFA	CFA	PFA

Der wichtigste Teil des Namenseintrags ist das Codefeld CD. Es enthält die Adresse des Maschinencodeprogramms, das der innere Interpreter aufzurufen hat. Die Adresse des Codefeldes heißt Codefeldadresse CFA.

Ein übersetztes (compiliertes) Forth-Programm besteht (fast ausschließlich) aus aufeinanderfolgenden Codefeldadressen.

Aus dem Paramterfeld (ab Parameterfeldadresse PFA) kann das Maschinencodeprogramm während seines Ablaufs die zum betreffenden Namen gehörigen Parameter lesen.

Sind z.B. mehrere Konstanten definiert, so steht in den Codefeldern ihrer Nameneinträge stets die gleiche Adresse, nämlich die Startadresse des Maschinenprogramms "Stapele Konstante"; der Wert der betreffenden Konstanten steht als Paramater im Parameterfeld des jeweiligen Namenseintrags. Das Maschinenprogramm "Stapele Konstante" braucht nur diesen einzigen Parameter.

Über das Linkfeld LK sind die Namenseinträge die miteinander verkettet.

Wenn ein Forthwort aus der Quellcodeform in den Zwischencode übersetzt werden soll, untersucht der Compiler das Wörterbuch von hinten nach vorne (d.h. von den jüngsten zu den ältesten Namenseinträgen); dabei findet er im Linkfeld jedes Namens die Anfangsadresse des nächstälteren Namenseintrags (das Linkfeld des ältesten Namenseintrags enthält 0).

Der Name und seine Länge sind im Namensfeld (ab NFA) gespeichert. Das Längenbyte LG enthält außer der Namenslänge (in Bytes) noch weitere Informationen (Immediate-Bit und Ungütigkeits-Bit); das oberste Bit von LG ist immer gesetzt, damit beim Rückwärtslesen des Namens sein Anfang erkannt werden kann. (Dies wird z.B. beim Trace benötigt, der ja aus der Codefeldadresse den Namen ermitteln muß).

Gewöhnlich übersetzt der Compiler ein druckbares Forthwort in eine Codefeldadresse. Es gibt jedoch Sonderfälle in denen nicht ein Wort des Quellcodes genau einer Codefeldadresse entspricht.

Einige Beispiele:

a) Einer eingegebenen Zahl wird automatisch die Codefeldadresse von LIT vorangestellt. Durch diese Maßnahme wird erreicht, daß der Compiler die Zahl selbst in Binärform in das erzeugte Programm (hinter dem LIT-Auf ruf) speichern kann. Das Maschinenprogramm von LIT kopiert das auf den LIT-Aufruf folgende Doppelbyte einfach auf den Stapel und braucht sich nicht mehr mit der Dekodierung der eingegeben Zahl aufzuhalten. Es muß lediglich noch dafür sorgen, daß der Interpreter das Doppelbyte überspringt.

b) Worte, die den Compiler steuern, werden während des Compilierens sofort ausgeführt. Diese Worte erkennt der Compiler daran, daß im Längenbyte ihres Nameneintrags das "Immediate-Bit" gesetzt ist. Der Anwender kann den Compiler

erweitern umd modifizieren; um zu erreichen, daß seine eigenen den Compiler erweiternden Worte vom Compiler nicht übersetzt sondern sofort ausgeführt werden, muß er in ihrem Namenseintrag das Immediate-Bit setzen; zu diesem Zweck steht die Funktion IMMEDIATE zur Verfügung.

c) Forthworte, die mit dem Zeichen" enden, z.B.." veranlassen den Compiler im allgemeinen, folgende Informationen ins erzeugte Programm zu legen:

— Die Codeadresse der zugehörigen Laufzeitfunktion, deren Name durch Einklammern des Funktionsnamens gegeben ist, also z.B. (.")

— Den Zeichenstring mit vorangestelltem Längenbyte

Wird später die kompilierte Funktion aufgerufen, so verarbeitet die Laufzeitfunktion — z.B. (.") — den auf ihren Aufruf folgenden Zeichenstring und veranlaßt den Interpreter, die Verarbeitung erst hinter dem String fortzusetzen.

d) "Strukturworte" wie BEGIN, UNTIL usw. werden im Prinzip folgendermaßen verarbeitet:

Findet der Compiler das Wort BEGIN, so legt er im erzeugten Programm überhaupt nichts ab; er legt aber die augenblickliche Adresse HERE (also die Adresse, ab der die nächste Codefeldadresse gespeichert werden soll) auf den Stapel.

Dort bleibt die Adresse liegen, bis der Compiler im Lauf seiner weiteren Arbeit auf das zugehörige Wort UNTIL trifft. An dieser Stelle legt der Compiler ins erzeugte Programm die Codefeldadresse OBRANCH und danach die relative Rücksprungadresse, die (mit Hilfe der Funktion BACK) aus dem obersten Stapeleintrag und der augenblicklichen Ablageadresse errechnet wird.

Wird später die erzeugte Funktion aufgerufen, so veranlaßt die Funktion OBRANCH einen bedingten Rücksprung des Interpreters entsprechend der hinter dem OBRANCH abgelegten relativen Rückwärtssprungadresse.

Der Forth-Compiler kommt mit den Sprungfunktionen BRANCH und OBRANCH aus; alle aus Strukturworten erzeugten Sprünge werden durch diese beiden Funktionen realisiert.

Während des Compilierens werden Strukturprüfungen durchgeführt.

Bestimmte Forthworte dürfen nur gruppenweise angegeben werden; z.B. darf der Anwender das Wort LOOP nicht gebrauchen, ohne vorher DO eingegeben zu haben. Verstöße gegen diese Regel (sowie auch falsche Schachtelung usw.) werden üblicherweise als "Strukturfehler" bezeichnet. Folgendes Verfahren, erlaubt dem Compiler Strukturfehler zu erkennen:

a) Am Anfang jeder Wortdefinition (d.h. wenn der Compiler den Doppelpunkt verarbeitet) wird der augenblickliche Inhalt des Stapelzeigers in der Anwendervariablen CSP ("current Stack position") sichergestellt. Am Ende (also beim Semikolon) muß der Stapelzeiger wieder den gleichen Wert haben; sonst kommt die Fehlermeldung "Strukturfehler", und das Ungültigkeits-Bit im Namenseintrag wird nicht gelöscht.

b) Das erste Wort jeder Wortgruppe veranlaßt den Compiler ein Kennzeichen (1, 2, 3 oder 4) für die eröffnete Wortgruppe auf den Stapel zu legen. Folgt später ein Wort, das nur in einer eröffneten Wortgruppe auftreten darf, so prüft der Compiler, ob die richtige Wortgruppe aktuell eröffnet ist. (Wenn nicht, kommt die Fehlermeldung " paßt nicht"). Findet der Compiler ein Wort, das das Ende einer Wortgruppe kennzeichent, (wie UNTIL oder END), so wird (falls das Wort paßt), der Stapeleintrag wieder entfernt.

Dieses (auch bei anderen Compilern übliche) Verfahren hat den Vorteil, schon zur Compile-Zeit alle erkennbaren Fehler auszuschalten, so daß das erzeugte Programm später (wenn es abläuft) nicht mehr durch die zeitraubenden Strukturfehler-Abfragen belastet wird.

Folgende Kennzeichen werden verwendet:

DO .. LOOP und DO .. +LOOP verwenden als Kennzeichen die Zahl 3
BEGIN .. AGAIN und BEGIN .. UNTIL verwenden 1
IF .. ENDIF und IF .. ELSE .. ENDIF verwenden 2
CASE: .. ;CASE verwenden 4

BEGIN .. WHILE .. REPEAT arbeiten etwas komplizierter:

BEGIN stapelt zunächst 1,

WHILE wirkt wie IF , erhöht aber das von IF gesetzte Kennzeichen von 2 auf 4, damit die Wortgruppe nicht mit ENDIF abgeschlossen werden kann;

REPEAT erzeugt einen unbedingten Rücksprung wie AGAIN (nachdem seine eigene Rücksprungadresse und das aktuelle Kennzeichen 4 kurzfristig sichergestellt und das darunterliegende von BEGIN gestapelte Kennzeichen 1 geprüft und beseitigt wurde), vermindert dann das Kennzeichen 4 wieder um 2 (denn das CFA in der Definitionszeile von REPEAT wirkt wie 2 —) und beschließt seine Arbeit genau wie ENDIF (wobei auch geprüft wird, ob das soeben verminderte aktuelle Kennzeichen nun 2 ist).

Anmerkung:

Die Wortgruppe CASE: .. ;CASE kann dasselbe Kennzeichen (nämlich 4) verwenden, das in BEGIN .. WHILE .. REPEAT zwischenzeitlich gebraucht wird, weil im Falle einer fehlerhaften Verschachtelung der beiden Wortgruppen auf jeden Fall eine der Strukturfehlerprüfungen ansprechen würde.

Anhang D

MC-12-Befehle

Um den Zugriff auf das MC-12-System von FORTH aus zu vereinfachen, wurde eine Reihe von Zusatzfunktionen in das PC-FORTH-System aufgenommen.

Generell besteht natürlich die Möglichkeit, alle MC-12-BASIC-Kommandos auch innerhalb von PC-FORTH zu verwenden, da BASIC-Unterprogramme von FORTH aufgerufen und Parameter übergeben werden können. Der MC-12-Benutzer kann also ohne Schwierigkeiten FORTH-Worte für MC-12-Funktionen definieren, die kurze BASIC-Unterprogramme aufrufen (vgl. Kapitel 4).

Für zeitkritische Funktionen ist diese Methode natürlich ungeeignet. Deshalb wurde PC-FORTH um Funktionen erweitert, die den schnellen Zugriff auf die Hardware-Elemente und die Meßwertspeicher des MC-12 ermöglichen.

Um überflüssige Konversionen zu vermeiden, werden alle (auch die in Puffern gespeicherten) Meßwerte als vorzeichenfreie Dualzahlen behandelt, wie sie die Hardware des MC-12-Systems abliefert — unabhängig davon, ob sich das MC-12-System in der bipolaren oder der unipolaren Betriebsart befindet.

Gleichzeitig wurden jedoch Konversionsroutinen zur Verfügung gestellt, die Meßwerte unter Angabe des Meßbereichs (RANGE) in eine vorzeichenbehaftete Ganzzahl konvertieren, die dem Meßwert in der Einheit mV entspricht. Diese Funktionen erlauben es dem FORTH-Programmierer genau da zu konvertieren, wo es notwendig ist.

8-Bit-Meßwerte liegen im Bereich zwischen 0 und 255.0 entspricht dem Minimum im jeweiligen Meßbereich, 255 dem Maximum. Bei bipolarer Messung liegt der Nullpunkt also bei 128. Wird mit 11 Bit gemessen, so liegen die Werte zwischen 0 und 65536 (genauer &FFE0), das höherwertige Byte entspricht dabei genau dem 8-Bit-Wert. Der Nullpunkt einer bipolaren Messung liegt also bei 32768 (&8000).

Mittels der Konversionsfunktionen ''MV' und "DMV' können die Werte jederzeit in Millivolt umgerechnet werden.

Die meisten der folgenden FORTH-Worte haben die gleiche Funktion wie die entsprechenden MC-12-BASIC-Befehle. Sie können also bezüglich der Auswirkung des jeweiligen Befehls zusätzlich die MC-12-Anleitung zu Rate ziehen. Die Befehle sind in der folgenden Beschreibung nach Sachgruppen geordnet.

Zugriff auf die MC-12-Hardware

MCON (-)
Schaltet das MC-12-System ein, wobei eine Zeitverzögerung eingebaut ist, die dafür sorgt, daß sich die Hardware des MC-12 stabilisiert.

MCOFF (--)
Schaltet das MC-12-System aus.

SWITCHON (a -)
Schaltet den Schalter mit der Nummer a ein. Dabei kennzeichnen die Nummern 1 bis 4 die CMOS-Schalter 1 bis 4 des MC-12, die Nummern 5 und 6 die Relays RMTO und RMT1 des CE-150.

SWITCHOFF (a -)
Schaltet den Schalter mit der Nummer a aus. Bedeutung der Nummern wie bei SWITCHON.

OUTCHA (a n -)
Stellt am Analogausgang a den Wert n ein. Dabei liegt n zwischen 0 und 255, a kennzeichnet die Nummer des Ausgangskanals.

SETRANGE (a n -)
Stellt am Eingang a den Meßbereich n ein. Die Eingangskanäle des MC-12 sind von 1 bis 5 numeriert, die Meßbereiche von 1 bis 11. Dabei ist Bereich 1 der unempfindlichste Bereich, 11 der empfindlichste.

RANGE (a - n)
Liefert die Nummer n des augenblicklich eingestellten Meßbereichs am Kanal mit der Nummer a.

Zugriff auf die Meßwertspeicher

BUFNUM (- n)
Liefert die Anzahl der Pufferspeicher.

BUFLEN (- n)
Liefert die Länge der Pufferspeicher.

?DBUF (- f)
Liefert eine 1, falls die Pufferspeicher 16-Bit-Werte enthalten (
(mit DBUFINIT initialisiert wurden), andernfalls eine 0.

BUFOPEN (a n -)
Löscht den Puffer a und belegt ihn mit dem Meßbereich n.

BUFRANGE (a - n)
Liefert den Meßbereich, der in Puffer a gespeicherten Werte.

BUFREAD (al a2 - n)
Liefert den in Puffer al an der Position a2 gespeicherten Wert n, je nach ?DBUF im Format von 8- oder 11-Bit-Meßwerten.

BUFWRITE (al a2 n -)
Schreibt in den Puffer al an die Position a2 den Wert n, wo- (
bei dieser je nach ?DBUF ein 8- oder 16-Bit-Wert sein kann.

Weitere Hilfsmittel

2@: (- d)
wird in der Form

2@: "basicausdruck"

gebraucht (vgl. "@:") und liefert den Wert des BASIC-Ausdrucks als Doppelzahl.